UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: Navegación Indoor 2D/3D en Dispositivos Móviles
Autores: Borré Claudio, Appathie Luciano
Director: Gordillo Silvia
Codirector: Challiol Cecilia
Asesor profesional: -
Carrera: Licenciatura en Sistemas

Resumen

La orientación en espacios que no conocemos es parte de la vida diaria. Habitualmente, como parte de sus actividades las personas ingresan en diferentes espacios tales como edificios, fábricas, aeropuertos, shoppings, etc. Donde desconocen su distribución o nunca antes habían visitado con anterioridad. En la medida que los espacios se vuelven más complejos y extensos, la tarea de encontrar el lugar al cual se dirigen se vuelve más compleja e insume más tiempo. Otros aspectos como las limitaciones de movilidad (sea debido a una limitación de la persona, o bien a los objetos que transporta) agregan nuevos aspectos de complejidad en la búsqueda del camino que una persona debe seguir para llegar a su destino.

El objetivo del trabajo es el diseño e implementación de un mecanismo para la navegación en interiores utilizando códigos QR para establecer el posicionamiento. El guiado y presentación de la información es realizado a través de dispositivos móviles. El énfasis de este trabajo será el desarrollo de mecanismos alternativos a los tradicionales planos de planta para el guiado de las personas en interiores, con el objetivo de lograr una mejor orientación del usuario.

Palabras Claves

Navegación en Interiores, Posicionamiento en interiores, Localización indoor, LBS, Dispositivos Móviles, Códigos QR, HTML5, WebGL.

Trabajos Realizados

Diseño e implementación de un sistema de navegación indoor que permite al usuario visualizar (a través de una interfaz Web) el lugar donde se encuentra navegando. En particular, se desarrolló un prototipo que representa el plano del primer piso de la Facultad de Informática de la UNLP. El mismo simula la lectura de códigos QR, permitiendo al usuario navegar dentro del mapa.
Durante la navegación, el usuario puede seleccionar el lugar a donde desea dirigirse, visualizar galerías multimedia de las estructuras navegables, hacer Zoom.
Diseño e implementación de una interfaz que permite la edición de planos.

Conclusiones

En primer lugar, la implementación de posicionamiento indoor, puede presentar problemas de precisión, si la infraestructura existente y los dispositivos que se utilizan no cuentan con requerimientos específicos. En particular, el posicionamiento vía códigos de barra es claramente el más accesible para los usuarios, pero representa un mecanismo de sensado indirecto. Esta característica, podría afectar la navegación del usuario.
Por otro lado, la variación constante de las librerías de visualización, hace que la aplicación deba estar preparada para evolucionar acorde con estos cambios.

Trabajos Futuros

Contemplar otros mecanismos de sensado que permitan un posicionamiento automático y no explícito.

Investigar la utilización de los algoritmos 3D, de manera de optimizar la implementación de una interfaz en 3 dimensiones.

Mejorar la herramienta de construcción de planos.

Fecha de la presentación: Abril 2013

ÍNDICE GENERAL

Índice de Contenidos

INDICE DE FIGURAS

Índice de Figuras

1.Introducción

El conocer en qué lugar estamos ha sido una de las grandes brechas que la computación móvil ha intentado resolver en los últimos años. Los sistemas de posicionamiento global (GPS), han evolucionado de tal manera que hoy en día casi no existe dispositivo móvil que no los contenga. Por otra parte, la localización en interiores es uno de los campos más prometedores en la computación móvil; los servicios de información basados en la localización de personas u objetos están sobresaliendo y siendo llamados a revolucionar la forma de rentabilizar las inversiones en infraestructuras. Estos sistemas permiten desarrollar aplicaciones basándonos en el posicionamiento de las personas u objetos en tiempo real.

No obstante, la problemática de la localización en interiores ha sido objeto de estudio e investigación en los últimos años. Aunque existen diversos sistemas y métodos para resolverlos, todavía no se han conformado sistemas de navegación análogos tan populares como los existentes para GPS. Las razones de esta problemática se extienden tanto a lo técnico como a lo económico, técnico porque la localización en interiores tiene retos tecnológicos muy superiores a los planteados en espacios exteriores, económico porque la gran mayoría de los sistemas propuestos para solucionar los problemas de la localización en interiores utilizan una gran cantidad de infraestructura (puntos de control, sensores, etc).

1.1. Motivación

En la actualidad, las aplicaciones de mayor interés comercial provienen del posicionamiento contextual (context-aware), que permite el desarrollo de aplicaciones que reaccionan ante los cambios de contexto de los usuarios.

El proyecto planteado tiene como objetivo el diseño e implementación de un sistema de posicionamiento basado en interiores. Por cuestiones de costo y simplicidad, el prototipo se implementó utilizando la lectura de códigos QR. Los puntos principales que se persiguen son los siguientes:

- Definir y diseñar una arquitectura de sistema que permita la implantación del sistema en la Facultad de Informática. Este entorno constará sólo con la planta baja de la misma.

- Implementar el software necesario para el sistema de navegación en interiores. Esto incluirá, entre otros, un programa que ejecutará el usuario (GUI), simulando la lectura de un código QR y cuya intención sea localizarlo en la planta, ofreciendo la posibilidad de navegar desde su ubicación hasta un destino seleccionado.

1.2. Estructura del proyecto

Esta memoria está estructurada en 8 capítulos, según se indica a continuación:

- En el Capítulo 1 se presenta la navegación en interiores y se definen los objetivos y la la estructura del presente proyecto.

- En el Capítulo 2 se presenta un marco teórico sobre los sistemas basados en localización y se hace un breve estudio de los diferentes algoritmos de localización existentes para los dispositivos móviles.

- En el Capítulo 3 presentamos un breve estudio sobre la infraestructura móvil y los distintos sistemas operativos que podemos encontrar.

- El Capítulo 4 explica en profundidad todas las herramientas que se utilizaron para el desarrollo del prototipo.

- En el Capítulo 5 se describen el comportamiento que deberá de implementar la aplicación para cumplir con su objetivo.

- En el Capítulo 6 se presenta el modelo de la aplicación.

- En el Capítulo 7 se presenta la aplicación desarrollada ejemplificando cada una de sus funcionalidades.

- En el Capítulo 8 se describe el desarrollo de la interfaz 3D.

- En el Capítulo 9 se describen las conclusiones, resultados obtenidos y líneas futuras planteadas para mejorar el software de localización.

- El Capítulo 10 detalla toda la bibliografía utilizada para el desarrollo de este proyecto.

- Por último, el Apéndice A presenta en tres partes como fué implementada la aplicación con el respectivo código.

2.Guías Móviles

2.1. Sistemas de Información geográfica (GIS)

Geographic **I**nformation **S**ystem (Sistema de Información geográfica) es un sistema diseñado para capturar, manipular, analizar y presentar todo tipo de datos geográficos. En términos simples, GIS es la fusión de Cartografía, análisis estadístico y tecnología de bases de datos. **[1]**
El término GIS describe cualquier sistema de información geográfica que integra, almacena, edita, analiza, comparte y muestra la información geográfica para informar la toma de decisiones. Aplicaciones GIS son herramientas que permiten a los usuarios crear consultas interactivas, analizar la información espacial, editar datos en mapas y presentar los datos de todas estas operaciones.

En los últimos años, los sistemas GIS han asistido a numerosas aplicaciones destinadas a editar cartografía en entornos web, ejemplos de estas son Google Maps **[2]**, Google Earth **[3]**, Bing Maps **[4]**, StreetView **[5]**, entre otras. Estos entornos web dan a las personas acceso a enormes cantidades de datos geográficos, algunos de ellos utilizan software que a través de una API, permiten a los usuarios crear aplicaciones personalizadas.

Las tecnologías GIS trabajan con información digital, para la cual existen varios métodos utilizados en la creación de datos digitales. El método más utilizado es la digitalización, donde a partir de un mapa impreso o con información tomada en campo se transfiere a un medio digital por el empleo de un programa de Diseño Asistido por el Ordenador (DAO o CAD) **[6]** con capacidades de georreferenciación.
Dada la amplia disponibilidad de imágenes (tanto de satélite y como aéreas), la digitalización por esta vía se está convirtiendo en la principal fuente de extracción de datos geográficos. Esta forma de digitalización implica la búsqueda de datos geográficos directamente en las imágenes aéreas en lugar del método tradicional de la localización de formas geográficas sobre un tablero de digitalización.

Existen dos formas de almacenar los datos en un GIS:

Vectorial: Las características geográficas se expresan con frecuencia como vectores, manteniendo las características geométricas de las figuras. Los GIS que se centran en el manejo de datos en formato vectorial son más populares en el mercado.**[7]**

Raster: Divide el espacio en celdas regulares donde cada una de ellas representa un único valor. Son muy utilizados en estudios que requieran la generación de capas continuas, necesarias en fenómenos no discretos; también en

estudios medioambientales donde no se requiere una excesiva precisión espacial (contaminación atmosférica, distribución de temperaturas, localización de especies marinas, análisis geológicos, etc.).**[8]**

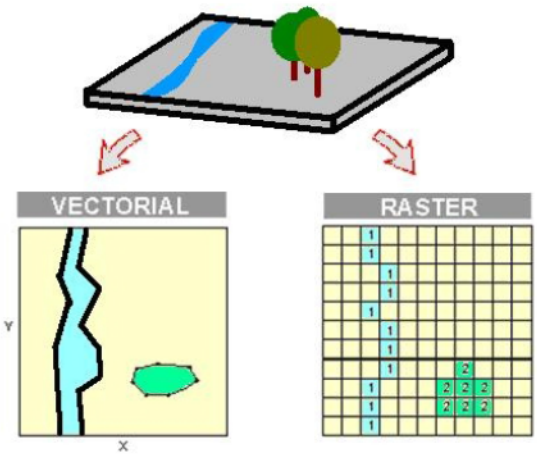


Figura 1. Interpretación vectorial (izquierda) y raster (derecha)[1]

2.2. Sistemas basados en Localización (LBS)

En la actualidad, GIS están teniendo una fuerte implementación en los Sistemas Basados en Localización (LBS), debido al abaratamiento y masificación de la tecnología GPS integrada en dispositivos móviles (teléfonos móviles, PDAs, ordenadores portátiles, etc). Los sistemas LBS permiten a los dispositivos con GPS mostrar su ubicación respecto a puntos de interés fijos (hoteles, restaurantes, cajeros, estaciones de servicio) o móviles (amigos, colectivos, etc) o para transmitir su posición a un servidor para la visualización u otro tipo de tratamiento. **[9]**

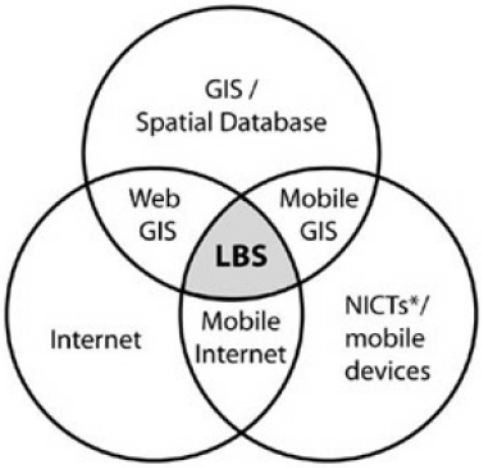


Figura 2. LBS como una intersección de tecnologías[2]

[1]http://1.bp.blogspot.com/-M9WnYTj9SPY/UIRi1i1yYrI/AAAAAAAAABY/1SFLMoP8OlM/s1600/modeloss.jpg
[2]http://www.e-cartouche.ch/content_reg/cartouche/LBSbasics/en/image/lbs_roots_circle150dpi.jpg

Para la gran mayoría de las aplicaciones, lo que realmente se desea es la ubicación dentro de un edificio o de un área, o la ubicación con respecto a otras personas u objetos, ya sea en movimiento o parado. La mayoría de los sistemas de posicionamiento permiten la determinación autónoma de posición, sin embargo, esta información debe ser comunicada por un mecanismo independiente con el fin de ser compartida. El posicionamiento de datos es sólo un aspecto a ofrecer para los servicios LBS.

Las aplicaciones típicas LBS buscan proveer servicios geográficos en tiempo real. Algunos ejemplos típicos de esto son servicios de mapas callejeros (Ej. obtener un mapa de una zona Google Maps), enrutamiento (Ej. Establecer una ruta entre una dirección origen y una dirección destino - Servicio enrutamiento Tom Tom). Existen básicamente dos escenarios para las aplicaciones basadas en localización: **Un usuario puede necesitar de un servicio de información** o el centro de administración puede requerir **rastrearlo -hacer tracking - en tiempo real. [10]**

En cualquiera de los dos escenarios, a través de algún mecanismo (Ej. dispositivo de posicionamiento integrado con el móvil) se determina la posición actual del usuario; esta ubicación junto con otros parámetros relevantes, es transmitida a un centro de procesamiento. Allí, los requerimientos de servicio son analizados por una infraestructura apoyada en sistemas de información geográfica para poder entregar la respuesta al usuario. Hay dos modos de trabajo, sistema LBS activo y pasivo.

Un sistema LBS activo esta enfocado principalmente en usuarios móviles particulares con el fin de proveer a ellos información de servicios. Los sistemas LBS pasivos generalmente son diseñados para clientes empresariales que requieren **administrar recursos móviles** (Ej. conocer en tiempo real la ubicación de la fuerza de venta en campo o hacer seguimiento de flotas de vehículos) y **soportar toma de decisiones** (Ej. definir estrategias de geo-marketing).

2.3. Indoor Positioning System (IPS)

Un sistema de posicionamiento en interiores (IPS) es un término utilizado para una red de dispositivos que se utilizan para localizar de forma inalámbrica objetos o personas dentro de un edificio.

En lugar de utilizar los satélites, un IPS cuenta con anclajes cercanos (nodos con una posición conocida). Los Sistemas de Navegación por GPS generalmente no son adecuados para ser utilizados en interiores, esto es debido a que las microondas se atenúan y dispersan por techos, paredes y otros objetos. Sin embargo, con el fin de hacer que la señal de posicionamiento se pueda conseguir en todas partes, la integración entre el GPS y el posicionamiento en interiores puede ser realizada mediante el uso de cualquier tecnología inalámbrica. **[11]**

2.3.1. Indoor vs Outdoor

Los datos que subyacen a los sistemas tradicionales de información geográfica (GIS) por lo general no contienen ni los detalles ni los sistemas de coordenadas para satisfacer las necesidades de los usuarios en un ambiente interior. Los Sistemas Basados en Localización (LBS) requieren detalles de georeferenciación más especializados para satisfacer las necesidades de los interiores. No es suficiente la georreferenciación de un edificio si las posiciones de los usuarios y de otros objetos en el interior del edificio son relevantes. Los objetos estáticos se utilizan como puntos de referencia (referencias espaciales), y las relaciones entre estos objetos son cruciales para la representación simbólica de un sistema. **[12]**

Consideremos como ejemplo un centro comercial donde el nivel de detalle de mapeo de la ubicación del modelo subyacente determina qué funcionalidad puede ser proporcionada a la aplicación. Por ejemplo, si la aplicación se utiliza al aire libre (outdoor), para navegar entre puertas, es suficiente para volver a lugares de la granularidad de edificios, mientras que en el interior de una habitación del edificio, el usuario está en un nivel más fino de granularidad (tal como los espacios entre sillas y/o estantes). Como resultado, la información geográfica tradicional no es lo suficientemente detallada para satisfacer las necesidades del usuario en un ambiente de interior.

Un LBS necesita guiar a un usuario desde una ubicación actual a un destino específico. La navegación de interiores puede parecer diferente a la navegación para automóviles (en la red de calles). Considerando que la navegación para automóviles es un proceso relativamente uniforme, que sigue la geometría de la carretera; los usuarios móviles caminan dentro de un edificio, por otro lado, son menos predecibles.

En Interiores, los usuarios de móviles no tienen un conjunto bien definido de vías, ya que pueden vagar en prácticamente cualquier espacio libre en el medio ambiente, mientras que los coches se circunscriben principalmente a circular sólo en la red de carreteras.

Sin embargo, los lugares de interiores no limitan a los usuarios móviles a un cierto nivel de acuerdo con la disposición del lugar, que puede ser modelado (explícitamente con los modelos topológicos) dentro del sistema de navegación en interiores. Los usuarios móviles pueden caminar hacia adelante, hacia atrás, y pueden cambiar el sentido de su caminar en cualquier momento, o se pueden mover arriba o abajo. Los desplazamientos verticales deben ser tenidos en cuenta debido a que los usuarios viajan arriba y abajo.

Explícitamente a los modelos topológicos, las técnicas utilizadas para el cálculo de una ruta entre cualquier origen y destino de interior, no son muy diferentes a las utilizadas en los caminos de computación al aire libre en la red vial.

Un sistema de navegación para peatones incluye lo siguiente:

- Un módulo de entrada, para la entrada de un lugar de partida y un destino del usuario.
- Una base de datos, con datos de ruta que representan la posición y conexión de cada ruta que consiste en una red de ruta (un mapa).
- Un módulo de cálculo de ruta para calcular una ruta desde el lugar de partida hasta el destino haciendo referencia a la base de datos de ruta.
- Opcionalmente, una base de datos histórica para almacenar los datos de punto de referencia para la confirmación de los peatones.
- Opcionalmente, un hito que representa una señal del lugar de salida, el destino y la ruta.
- Opcionalmente, un módulo de selección de punto de referencia para seleccionar los datos de señal correspondientes a la ruta calculada por la unidad de cálculo de la ruta de la base de datos de puntos de interés.
- Una unidad de presentación para presentar una guía de ruta para el usuario mediante el uso de la ruta calculada y los datos históricos.

2.3.2. Tecnologías disponibles de posicionamiento indoor

Los sistemas de posicionamiento en interiores a través de infraestructuras inalámbricas gozan hoy en día de gran popularidad. El avance de la tecnología ha permitido el desarrollo de dispositivos móviles con gran poder de cómputo que permiten la utilización de algoritmos que antes estaban reservados para equipos fijos.

La utilización de estos algoritmos en dispositivos móviles produjeron la aparición de distintas aplicaciones en campos variados como sistemas de transportes, facturación dependiendo de la localización, computación ubicua, servicios de información dependientes de la posición o del seguimiento del usuario por mencionar algunas de ellas. Muchas de estas aplicaciones requieren de una precisión en la localización relativamente alta, tanto en exterior como en interiores. **[13]**

Diferentes tipos de localizaciones:

- **Localización física**: identifica un punto en un mapa, ya sea 2-D o 3-D. El más extendido son las coordenadas físicas de grados / minutos / segundos (DMS y UTM[3])

[3]Es un sistema de coordenadas basado en la proyección cartográfica transversa de Mercator, que se construye como la proyección de Mercator normal, pero en vez de hacerla tangente al Ecuador, se la hace tangente a un meridiano.

- **Localización simbólica**: describe la posición mediante una etiqueta que hace referencia a una posición descrita en un lenguaje natural.

Al momento de decidir entre los diferentes algoritmos de localización en interiores es necesario tener en cuenta diferentes aspectos del algoritmo utilizado:

- **Performance:** La métrica mas importante de la performance es la *precisión* del posicionamiento, es decir, el error de distancias entre el punto estimado de localización y la ubicación real del usuario.
  Otras métricas importante en la performance del algoritmo son el *tiempo de respuesta, capacidad, alcance* y *escalabilidad.*
  El tiempo de respuesta es el tiempo en obtener la localización, hay algoritmos que son muy precisos en la determinación de la ubicación pero el tiempo de respuesta es elevado y en un sistema interactivo esto puede ser una limitación muy importante.
  La capacidad se refiere a la cantidad de estimaciones de posiciones que un sistema puede determinar en una medida de tiempo.
  El alcance define los límites del espacio donde el algoritmo puede estimar la localización.
  La escalabilidad es la métrica que nos determina cómo responderá el sistema cuando la cantidad de información a procesar aumente.

- **Costo y complejidad:** el costo de un sistema de posicionamiento puede ser determinado por el costo de infraestructura, ancho de banda adicional, tolerancia a fallos y confiabilidad. El costo puede incluir el tiempo destinado a la instalación de los dispositivos y sensores, si un sistema de posicionamiento puede rehusar de una infraestructura de comunicación, o una parte de ella, el equipamiento, etc. También es necesario considerar como costo el tráfico de datos entre el sistema de posicionamiento y el dispositivo móvil a posicionar.

- **Requerimientos de la aplicación:** estos son determinantes al momento de la elección de la tecnología de posicionamiento. Los requerimientos incluyen *granularidad* , *performance* y *disponibilidad.*
  La granularidad puede ser dividida en granularidad temporal, la cual determina la cantidad de respuestas o posiciones que puede determinar un sistema de posicionamiento en un tiempo determinado, y la granularidad espacial determina el nivel de detalle de la información de localización.
  Performance y disponibilidad están relacionadas a los puntos anteriores, son determinadas por el tipo de aplicación a construir, qué características serán deseables y cuales fundamentales. Por ejemplo una aplicación puede necesitar un corto tiempo de respuesta y ser flexible en cuanto a la precisión del sistema de posicionamiento.

- **Seguridad**: La información de la localización debe estar disponible sólo para aquellos usuarios con acceso autorizado. Este punto representa la privacidad

del usuario. Existen sistemas de posicionamiento con un protocolo de seguridad embebido , y otros en el que esta característica no está implementada.

2.3.3. Algoritmos de localización

A continuación se describen diferentes algoritmos **[14]** de localización:

- **Tiempo de llegada (TOA) :** es el tiempo que transcurre desde que se emite la señal en el dispositivo a localizar hasta que llega al dispositivo de medida. La distancia se calcula en forma de una función euclidiana.

$$f_i(x) = c(t_i - t) - \sqrt{(x_i - x)^2 + (y_i - y)^2}$$

  (donde c es la velocidad de la luz). En general se precisan tres puntos de referencia para una localización en dos dimensiones. Se necesita una gran precisión en la sincronización y en el tiempo de referencia. Este tiempo puede ser medido usando diferentes técnicas como espectro ensanchado (DSSS) o Ultra Wide Band (UWB), además de otros algoritmos basados en TOA como vecino más cercano, o peso residual.
- **Tiempo de llegada diferencial (TDOA):** se basa en determinar la posición relativa del usuario midiendo las diferencias de tiempo de la señal recibida en distintos dispositivos, en lugar del tiempo de llegada absoluto. Para cada medida, en lugar de una distancia euclidiana, quedan expresadas como una función hiperbólica:

$$R_{i,j} = \sqrt{(x_i - x)^2 + (y_i - y)^2] + (z_i - z)^2} - \sqrt{(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2}$$

  donde (Xi; Yi; Zi) y (Xj; Yj; Zj) representan las señales recibidas y (X;Y;Z) las coordenadas del objetivo.

- **Tiempo de vuelo (RTOF):** este método consiste en medir el tiempo de vuelo de la señal desde el transmisor hasta el usuario y de vuelta. Requiere una sincronización relativamente alta. Este método tiene un error de precisión de unos pocos metros. El mecanismo es similar al utilizado por un radar.

- **Ángulo de llegada (AOA):** determina la posición conociendo la intersección de varias líneas de dirección. Si conocemos el ángulo incidente de las ondas y la localización, y la distancia entre las estaciones base que emiten estas señales, podemos establecer la posición del objeto. Con este sistema, basta con tres señales de medida para poder posicionarse en tres dimensiones y tan solo dos para un posicionamiento en dos dimensiones. El problema es que necesita un hardware complejo, incrementando el precio y la escalabilidad.

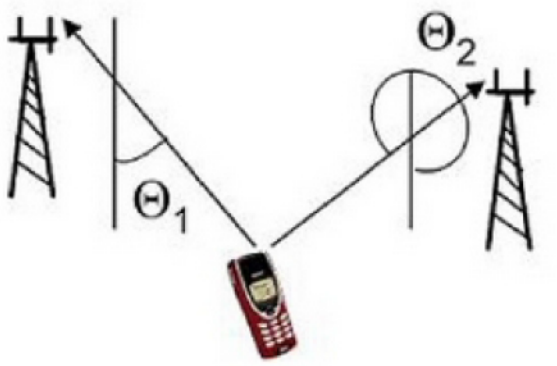


Figura 3. Ejemplo Escenario Ángulo de llegada[4]

- **Potencia de señal recibida (RSS):** Todos los métodos anteriores no muestran el rendimiento esperado si no tienen una línea de visión directa con el emisor de la señal. Utilizando la atenuación de la señal emitida nos da una precisión más alta en estas situaciones, pero a distancias cortas. Los algoritmos basados en la atenuación de la señal calculan el camino que ha recorrido la señal a través de la atenuación sufrida, por lo tanto la diferencia entre la potencia emitida y la recibida nos dará una idea de la distancia hasta el emisor.

- **Proximidad:** Simple y ligero, asume que el usuario esta en el area de antena de la que recibe mayor potencia. Cuanto mayor sea la densidad de antenas más preciso será el método.

2.3.4. Sistemas de posicionamiento en interiores

Analizaremos los distintos sistemas de posicionamiento desde el punto de vista del hardware que necesitan, se basan en los algoritmos descritos en la sección anterior. Describiremos algunas de las versiones comerciales **[15]** implementadas:

- **Active Bat:** utiliza una red de receptores ultrasónicos que reciben la señal del dispositivo y haciendo uso del algoritmo TOA determina la ubicación del usuario. La sincronización viene por parte de una señal de radio. Tiene una precisión de unos 10 centímetros en el 95% de los casos.

- **Cricket:** utiliza una red de sensores, señales de radio como medio de sincronizacion y señales ultrasonicas como medio de localización. Alcanza una precisión con un margen de error de 2 cms. Emplea TOA. Presenta una escalabilidad descentralizada y una alta privacidad, no obstante tiene altos requisitos de computación por la necesidad de precisos sincronismos.

- **Hexamite:** tiene tres tipos de transmisores: emisores, receptores y controladores. Utiliza el algoritmo de TDOA, no requiere sincronización. Obtiene una precisión de 1 cm pero necesita tener una alta densidad de

[4]http://www.slideshare.net/guest3014af17/sistemas-de-localizacin-basados-en-toa

sensores.

- **AHLos:** la principal ventaja de este sistema es su instalación dinámica, la red se despliega haciendo uso de infraestructura ya existente como GPS (funciona tanto en exteriores como interiores) para localizar sus propias balizas. Los nodos estiman su posición y distancia a los vecinos a partir de los nodos que ya conocen. Para la localización utiliza TDOA a través de una señal ultrasónica y una señal de radio. El mayor inconveniente son los costos de computación.

- **D-Locus:** la particularidad de este sistema es que funciona en el rango de señales acústicas dentro del espectro audible por el ser humano, el cual presenta unas buenas propiedades de atenuación y rango. Utiliza TOA y el sistema es bidireccional, es decir que todos los nodos son tanto emisores como receptores. Cuenta con una precisión de 1 cm. La mayor desventaja es la gran potencia de cálculo que requiere.

- **RADAR:** sistema desarrollado por Microsoft, basado en redes inalambricas IEE 802.11. RADAR mide la potencia y la relación señal-ruido y la emplea para posicionar el dispositivo en un mapa previamente muestreado a la base de fingerprints (huellas). Tiene un margen de error de unos 2-3 metros con solo cuatro puntos de acceso 802.11. Es un sistema de bajo coste ya que no requiere otro infraestructura más que los routers web que estén instalados.

- **PinPoint 3D-iD:** se colocan sensores que emiten secuencialmente señales mientras el dispositivo a localizar emite su propia señal. La medida de tiempo de estas dos señales se emplea para calcular la posición (Utiliza el algoritmo de tiempo de vuelo).

- **MotionStar magnetic tracker:** es un sistema de gran precisión espacial (1 mm y 0.1° de orientación de margen de error) y una gran resolución temporal (1 ms de refresco). Genera un sistema de seguimiento a través de pulsos continuos magnéticos, mide la posición y la orientación a través de una serie de antenas que reciben la respuesta en tres ejes espaciales. Su principal desventaja es que los sensores deben estar a una distancia de entre 1 y 3 metros del emisor.

- **UWB Ubisense:** utiliza señales de espectro muy ensanchado, tiene un alcance de unos 50 metros con poca densidad de balizas y una precisión de unos 20 cms.

- **Active Badge:** consiste en un móvil con un dispositivo de proximidad infrarrojo, este emite una señal de localización cada 10 segundos (programable) a sensores infrarrojos instalados en el edificio que mandan la información a un servidor central. Puede aportar por lo tanto, referencia

absoluta como simbólica. El mayor problema que presenta es que no funciona correctamente en presencia de luces fluorescentes o con luz directa del sol, ya que en ambos casos aumenta mucho el ruido en la banda de luz infrarroja.

- **Identificación por radiofrecuencia (RFID):** utiliza etiquetas (tags) activas, estas almacenan y recuperan datos a través de transmisiones electromagnéticas a circuitos integrados compatibles con radio frecuencia (RF). Estos circuitos con antenas pueden ser tanto pasivas (no requieren una fuente de alimentación interna y solo emiten señal cuando reciben una en un alcance menor a dos metros) como activas (requieren fuente de alimentación propia y pueden emitir señal periódicamente a mayor distancia o realizar tareas más complejas, son de mayor costo). Se basa en un algoritmo en tres dimensiones de agregación a través de la intensidad de la señal de radio. Estos sistemas suelen ser utilizados en sistemas de localización de equipajes en aeropuertos, o seguimiento de paquetes en almacenes industriales.

- **Smart Floor:** Consiste en una serie de sensores de presión en el suelo para capturar pisadas para localizar y seguir a peatones. Un problema que presenta es su escasa escalabilidad, dado que requiere un sensor por cada punto donde se localiza.

- **QR codes:** son códigos de barras en dos dimensiones los cuales pueden codificar información como una URL, y esta al ser leída desde un móvil obtiene la posición dentro del edificio. Al igual que Smart Floor su principal desventaja es la escalabilidad, ya que es necesario un QR code por cada punto referenciable, la ventaja de este método es su simplicidad y bajo costo.

3. Plataformas Móviles

En la actualidad los dispositivos móviles han aumentado su potencia de cómputo hasta acercarse a la capacidad de una PC, ya se pueden encontrar dispositivos con cuatro núcleos como el chip nVidia Tegra 3 **[16]**. Incluso los sistemas operativos comienzan a mostrar la tendencia de converger, Windows 8 **[17]** apunta a funcionar tanto en PCs como en dispositivos móviles o tabletas. Esto permite una libertad y una disponibilidad cada vez mayor de recursos para los desarrolladores.

El desarrollo de aplicaciones móviles conlleva una variedad de consideraciones de acuerdo al propósito y escenario para el que van a ser utilizadas. Hace algunos años se consideraba que desarrollar aplicaciones móviles significaba desarrollar una aplicación tradicional pero en "pequeño". Debemos empezar por conocer los tipos de aplicaciones que podemos implementar para los dispositivos móviles.

Figura 4. Tipos de aplicaciones móviles[5]

El desarrollo de aplicaciones móviles difiere del desarrollo de software tradicional en muchos aspectos, esto provoca que las metodologías usadas para estos entornos también difieren con las utilizadas en el software clásico. Esto es debido a que el software móvil tiene que satisfacer una serie de requerimientos **[18]** que lo hace más complejo:

- **Canal radio**: consideraciones tales como la disponibilidad, las desconexiones, la variabilidad del ancho de banda, la heterogeneidad de redes o los riesgos de seguridad deben tenerse especialmente en cuenta en este entorno de comunicaciones móviles.

- **Movilidad**: aquí influyen consideraciones como la migración de direcciones, alta latencia debido a cambios de estación base o la gestión de la información dependiente de localización. Sobre esta última, de hecho, se pueden implementar un sinfín de aplicaciones, pero la información de contexto asociada resulta muchas veces incompleta y varía frecuentemente.

- **Portabilidad**: la característica portabilidad de los dispositivos implica una serie de limitaciones físicas directamente relacionadas con el factor de forma

[5]http://blogs-images.forbes.com/fredcavazza/files/2011/09/app-web-native.jpg

de los mismos, como el tamaño de las pantallas, o del teclado.

- **Fragmentación de la industria**: la existencia de una considerable variedad de estándares, protocolos y tecnologías de red diferentes añaden complejidad al escenario del desarrollo móvil.

- **Capacidades limitadas de los terminales**: incluimos factores como la baja potencia de cálculo o gráfica, los riesgos en la integridad de datos, las interfaces de usuario poco funcionales en muchos aspectos, la baja capacidad de almacenamiento, la duración de las baterías o la dificultad para el uso de periféricos en movilidad. Factores todos que, por otro lado, están evolucionando en la dirección de la convergencia de los ultra portátiles (netbooks) con los dispositivos inteligentes constituyendo cada vez menos un elemento diferencial.

- **Usabilidad**: las necesidades específicas de amplios y variados grupos de usuarios, combinados con la diversidad de plataformas tecnológicas y dispositivos, hacen que el diseño para todos se convierta en un requisito que genera una complejidad creciente difícil de acotar.

- **Time-to-market**: en un sector con un dinamismo propio, dentro de una industria en pleno cambio, los requisitos que se imponen en términos de tiempo de lanzamiento son muy estrictos y añaden dificultad en la gestión de los procesos de desarrollo.

3.1. Categorización de las plataformas Móviles

Como todas las plataformas de software, estas son divididas en tres categorías: Licenciado, Propietario, y Código libre. **[19]**

3.1.1. Licenciado

Las plataformas licenciadas se venden a fabricantes de dispositivos para la distribución exclusiva en los dispositivos. El objetivo es crear una plataforma común de desarrollo de interfaces de programación de aplicaciones (APIs) que funcionan de manera similar a través de varios dispositivos con el menor posible esfuerzo necesario para adaptarse a las diferencias de hardware, aunque esto no es del todo cierto.

Las aplicaciones licenciadas que existen hoy son:

- **Java Micro Edition (Java ME):** Anteriormente conocido como J2ME, Java ME es por mucho, la plataforma de software más predominante en cualquier tipo de dispositivo móvil. Java ME combina una JVM (Máquina virtual de

Java) y un conjunto de APIs de Java para desarrollar aplicaciones para dispositivos de recursos limitados, como son los móviles. **[20]**

- **Windows Phone:** Anteriormente llamado Windows Mobile, es una versión licenciada y compacta del sistema operativo Windows, combinado con un conjunto de aplicaciones básicas para dispositivos móviles que se basa en el API Win32 de Microsoft. **[21]**

- **Entorno de ejecución de binarios para Wireless (BREW)**: BREW es una plataforma creada por Qualcomm(*) para dispositivos móviles. Las características técnicas incluyen: Una SDK(*), un modelo basado en APIs independientes que cubren toda la funcionalidad básica de un terminal, y finalmente es multiplataforma con soporte a gran cantidad de fabricantes. **[22]**

3.1.2. Propietario

Las plataformas propietarias son diseñadas y desarrolladas por los fabricantes de dispositivos para el uso exclusivo en sus dispositivos. No están disponibles para otros fabricantes de dispositivos de la competencia. Incluyen las plataformas:

- **BlackBerry:** BlackBerry mantiene su propia plataforma propietaria basada en Java, que se utiliza exclusivamente por sus dispositivos. **[23]**

- **Palm**: Palm utiliza tres diferentes plataformas propietarias. Su primer y más reconocida es la plataforma Palm OS basado en el lenguaje C/C++, la que fue inicialmente desarrollada para su línea de Palm Pilot. Con la aparición de teléfonos inteligentes, Palm creó webOS, que es una plataforma basada en Linux y su ejecución se hace totalmente en el marco del navegador WebKit. **[24]**

- **iPhone**: Apple utiliza una versión compacta de Mac OS X como plataforma para su línea de dispositivos iPhone, iPad e iPod, que ciertamente está basada en Unix. Actualmente esta en la versión 4G. Provee todas las herramientas y recursos para construir aplicaciones nativas con un buen soporte para multimedia y gráficos. **[25]**

- **Symbian**: La plataforma Symbian es el software de código abierto basado en Symbian OS, el sistema más utilizado operativo abierto para teléfonos móviles. La Fundación Symbian es una organización sin fines de lucro, fundada por Nokia, Motorola, Sony Ericsson, NTT DoCoMo, Texas Instruments, Vodafone, Samsung, LG y AT&T. **[26]**

3.1.3. Código libre

Plataformas de código abierto son las plataformas móviles que están disponibles gratuitamente para los usuarios que pueden descargar, modificar y editar. Estas plataformas son las más recientes en móviles, pero aún así cada vez están ganando más confianza con los fabricantes de dispositivos y desarrolladores.

- **Android**. El sistema operativo Android está basado en el kernel de Linux y ha sido desarrollado bajo el amparo de Google. En principio, nació como un sistema operativo para móviles, aunque hoy en día lo encontramos también en tablets y en Google TV. Desde su presentación en noviembre de 2007, estamos actualmente en la versión 4.2.
  Android ofrece un conjunto completo de software de código libre para dispositivos móviles: un sistema operativo, middleware y aplicaciones esenciales para móviles. Es uno de los más recientes SO para móviles del mercado, está basado en una versión modificada del kernel de Linux. Las aplicaciones para Android se programan en lenguaje Java y son ejecutadas en una máquina virtual diseñada para esta plataforma, que ha sido bautizada con el nombre de Dalvik, está optimizada para requerir poca memoria y diseñada para permitir ejecutar varias instancias de la máquina virtual simultáneamente, delegando en el sistema operativo subyacente el soporte de aislamiento de procesos, gestión de memoria e hilos. Permite un acceso fácil a prácticamente todas las funcionalidades hardware de los dispositivos en los que esté instalado. Provee a los desarrolladores de librerías la creación ágil y rápida de aplicaciones.
  Se distribuye bajo una licencia Apache versión 2, la cual, es una licencia de software libre creada por la Apache Software Foundation (ASF). La Licencia Apache permite al usuario del software la libertad de usarlo para cualquier propósito, modificarlo y distribuir versiones modificadas de ese software. La licencia Apache requiere la conservación del aviso de copyright y el disclaimer, pero no es una licencia copyleft, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.
  Android Market, es una tienda de aplicaciones de donde podemos descargar aplicaciones a nuestro sistema. Buena parte del éxito de la plataforma Android se debe a su aceptación por parte los programadores y la venta de aplicaciones en el Android Market, la tienda de aplicaciones de Android, en la que se permite a los desarrolladores publicar, sin ningún tipo de filtro, aplicaciones tanto gratuitas como pagas. **[27]**

- **OpenMoko**. Hace uso del núcleo Linux con un entorno gráfico implementado con el X.Org, GTK+. OpenMoko es desarrollado por First International Computer (FIC) bajo la licencia GPL (General Public License). **[28]**

- **MeeGo**: Esta plataforma es capaz de funcionar en teléfonos móviles, además de una variedad de dispositivos, como netbooks, sistemas en vehículos,

televisores, etc. MeeGo está basada en el gestor de paquetes RPM, y cuenta con el auspicio de la Linux Foundation. **[29]**

- **LiMo**: LiMo es una fundación conformada por varias empresas del mercado de los dispositivos móviles. Esta desarrollado para asistir en el diseño, desarrollo y despliegue de dispositivos móviles. Esta compuesto por un sistema modular, una arquitectura de plugins, sobre un SO con un entorno seguro en tiempo de ejecución. **[30]**

Podemos notar que la plataforma esta muy ligada al sistema operativo en plataformas propietarias. Las plataformas licenciadas, al igual que las de código libre, disponen de una especificación que permite a los fabricantes de dispositivos hacer equipos compatibles con tales plataformas.

3.2. Aplicaciones móviles

Ya comentamos en la sección anterior los principales Sistemas Operativos presentes en los dispositivos móviles de la actualidad. Al momento de desarrollar una aplicación, una de las decisiones más importantes es elegir que tipo de aplicación móvil se quiere desarrollar. **[31]**

3.2.1. Nativas

Este tipo de aplicaciones se ha hecho para ejecutarse en un dispositivo específico. Así, la mayor parte de las aplicaciones descargadas de la app store de apple son aplicaciones que sólo van a correr sobre iphone e ipad. Este tipo de aplicaciones se crean con distintos tipos de lenguajes. Las desarrolladas para iOS (el sistema operativo de iphone e ipad) lo hacen con los lenguajes: Objective C, C, or C++. Las aplicaciones desarrolladas para el sistema operativo Android lo hacen con lenguaje Java. Este tipo de aplicaciones corren de forma más eficiente sobre estos dispositivos ya que sus componentes están diseñados de forma específica para este sistema operativo.

wikipedia

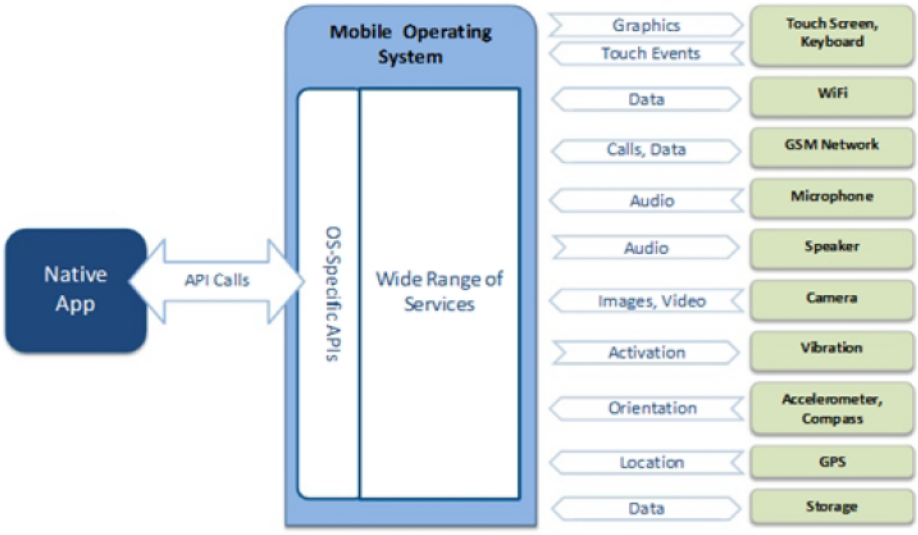


Figura 5. Aplicaciones Nativas[6]

Ventajas

- Las aplicaciones nativas tienen acceso total a las utilidades del sistema operativo del dispositivo: Dispositivos de almacenamiento, cámara, gps, acelerómetro, agenda, etc. Esto hace que la experiencia de usuario sea la más completa.
- Las aplicaciones nativas tienen la posibilidad de correr en modo offline, y su ejecución es superior a una aplicación híbrida o web.
- Por último es importante destacar que las aplicaciones nativas tendrán mucha más visibilidad ya que se distribuyen a través de la app store de los fabricantes.

Desventajas

- Cada plataforma tiene su propio lenguaje de programación, por lo cual la curva de aprendizaje es alta.
- Fragmentación del mercado, desarrollar para una plataforma específica puede significar quedar fuera de un gran porcentaje del mercado. El siguiente gráfico muestra la cuota del mercado de cada plataforma móvil, podemos ver que elegir desarrollar una aplicación para iOS significa quedar afuera del 70% del mercado.

[6]http://www.geospatialtraining.com/blog/wp-content/uploads/2012/03/hybrid4.png

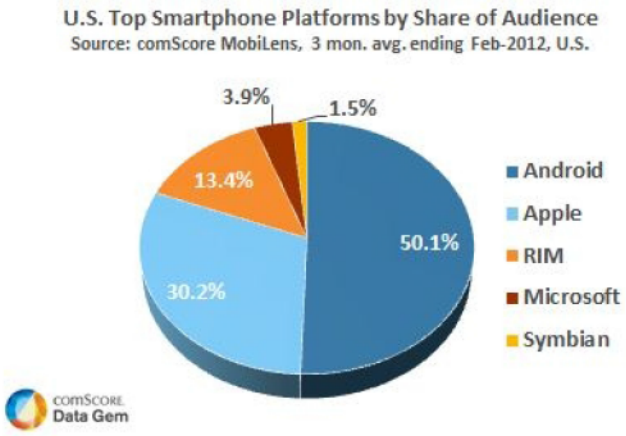


Figura 6. Mejores Plataformas para Smartphones U.S.[7]

- Fragmentación Interna, a su vez de la fragmentación del mercado existe una fragmentación interna dentro de una plataforma, por ejemplo, Android tiene un gran porcentaje en el mercado actual, pero existen muchas versiones en uso actualmente, desarrollar para la última versión tendrá ventajas como el acceso a las funciones más nuevas y optimizadas de la API de desarrollo pero dejará la aplicación fuera del alcance de los usuarios con versiones anteriores.

Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	1.9%
2.2	Froyo	8	7.6%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	44%
3.1	Honeycomb	12	0.3%
3.2		13	0.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	28.6%
4.1	Jelly Bean	16	14.9%
4.2		17	1.6%

Figura 7. Versiones de Android en el mercado[8]

3.2.2. Web

Las aplicaciones móviles web, a diferencia de las aplicaciones nativas se ejecutan en el navegador del teléfono, es por esto que al crear una aplicación web no hay que preocuparse por el sistema operativo que esté corriendo el dispositivo.

[7]http://www.winbeta.org/sites/default/files/newsimages/Smartphone-Platform-Share_February-2012-Data.jpg

[8]http://www.smart-gsm.com/blog/wp-content/uploads/2013/03/android-versions-march-2013.jpg

Se crea en un nivel de abstracción mayor, se pierde el acceso a muchos recursos del sistema (agenda, lista de llamadas, acelerómetro, gps, etc).

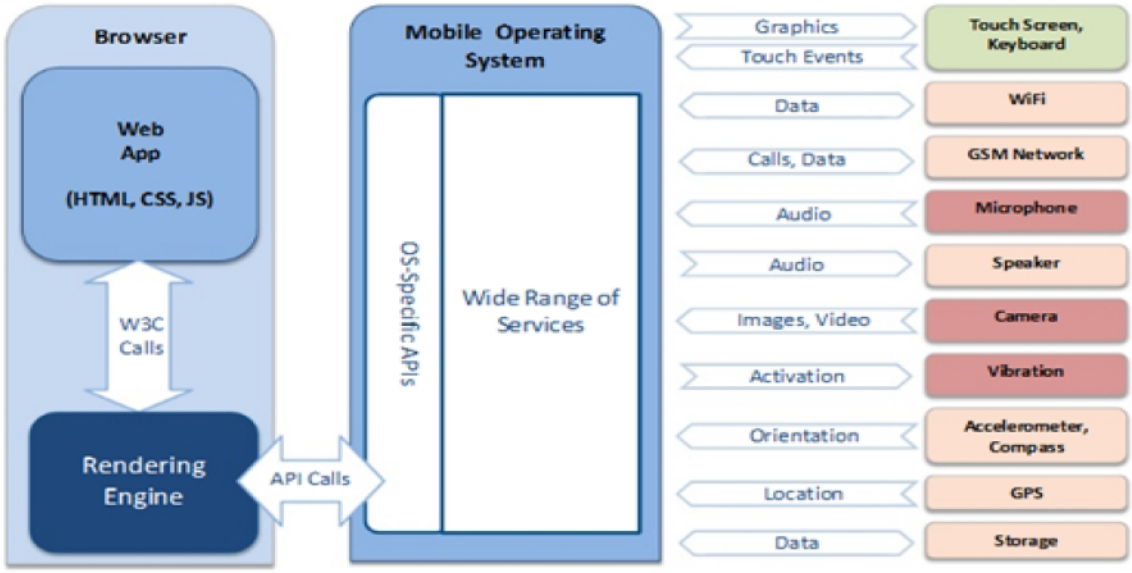


Figura 8. Aplicaciones Web[9]

Ventajas

- Al contrario que las aplicaciones nativas, las aplicaciones web se pueden ejecutar en múltiples dispositivos evitando así las complejidades de tener que crear varias aplicaciones.
- El proceso de desarrollo es más sencillo ya que emplean tecnologías ya conocidas como HTML, CSS y Javascript.
- Estas aplicaciones se pueden encontrar con los tradicionales buscadores.
- No necesitan de la aprobación para ser publicadas de ningún fabricante.

Desventajas

- Como desventajas tenemos que el acceso a los elementos del teléfono son limitados.
- Estas aplicaciones no se pueden vender en los marketplace.
- La velocidad de ejecución de las aplicaciones web dista mucho del desarrollo nativo.
- La aplicación necesita conexión a internet para funcionar.

3.2.3. Hibridas

Las aplicaciones híbridas unen lo mejor de los dos anteriores modelos. Este tipo de aplicaciones permite el uso de tecnologías multiplataforma como HTML, Javascript y CSS pero permite acceder a buena parte de los dispositivos y sensores del teléfono. Buena parte de la infraestructura es tipo web y la comunicación con los

[9]http://www.geospatialtraining.com/blog/wp-content/uploads/2012/03/hybrid5.png

elementos del teléfono se hace mediante comunicadores tales como phonegap. Un buen ejemplo de aplicaciones híbridas es Facebook. Se descarga de la app store y cuenta con todas las características de una aplicación nativa pero requiere ser actualizada ocasionalmente.

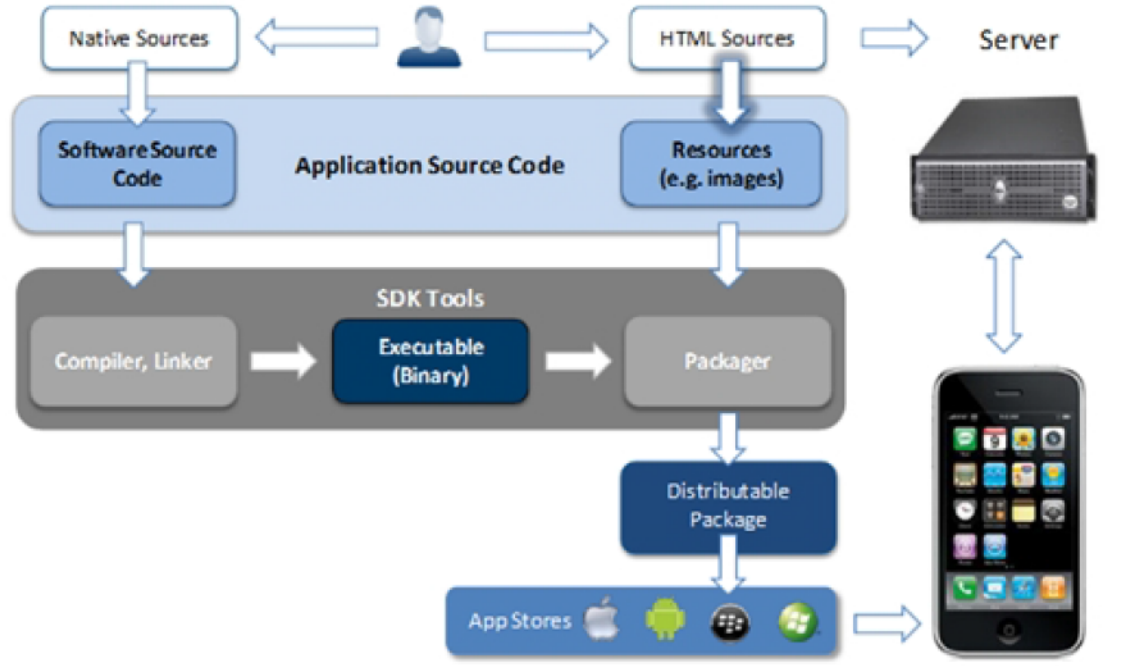


Figura 9. Aplicaciones Híbridas[10]

El proceso de desarrollo para este tipo de aplicaciones es algo más complicado. Al igual que para las aplicaciones nativas, el código una vez creado se compila a un ejecutable. Además, también como en las aplicaciones Web se genera código HTML, CSS y Javascript a ejecutar en un navegador. Ambos códigos se compilan para ser subidos mediante un paquete distribuible a la app store.

PhoneGap es una herramienta que nos permite este tipo de desarrollos, existen más herramientas similares, pero PhoneGap es una de las más utilizadas actualmente.

Nos permite crear un proyecto web, pero incluyendo una API propia de phonegap que nos da acceso a recursos del sistema como GPS, acelerómetro, etc. Una vez creada, phonegap compila el proyecto y crea una aplicación para cada sistema operativo elegido.

La ejecución se hace a través de un navegador del dispositivo, maximizado para que parezca una aplicación nativa, la velocidad de ejecución es mejor que una aplicación web pero sigue siendo más lenta que una aplicación nativa.

[10]http://www.geospatialtraining.com/blog/wp-content/uploads/2012/03/hybrid6.png

4. Herramientas utilizadas para el desarrollo

A continuación daremos una breve explicación de las tecnologías que hemos utilizado para programar nuestro prototipo

4.1. HTML5

Dentro del lenguaje HTML específicamente, las novedades que nos trae en su versión 5 son una serie de etiquetas útiles en la web actual, algo bien sencillo de aprender y de aplicar a los sitios web, así como el propio lenguaje de marcación es también sencillo de asimilar. Podemos clasificarlas en dos partes:

- Etiquetas que nos traen soporte a nuevas funcionalidades: es decir, aquellas que nos sirven para extender el HTML, dando soporte a asuntos como el vídeo o el sonido, lienzos donde diseñar dibujos, etc.

- Etiquetas que componen la web semántica: algunas etiquetas que realmente no proponen nuevas funcionalidades, sino que sirven para componer sitios indicando qué son los bloques de código de una web, en lugar de cómo se deben representar.

Para su uso es muy recomendable tener el navegador actualizado a su última versión, pudiendo de esta manera dar soporte a las etiquetas y funcionalidades que HTML5 nos provee **[32]**.

Nuevos elementos de formulario:

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas <DIV> y <SPAN>

- <AUDIO>: Para insertar sonido dentro de una web.
- <VIDEO>: Para insertar clips de vídeo.
- <EMBED>: Para embeber contenido externo de otro tipo, como el traído de diversos plugins que se comercializan actualmente o se comercializarán en el futuro.
- <SOURCE>: Permite especificar varias fuentes diferentes cuando se insertan elementos en <AUDIO> y <VIDEO>.
- <TRACK>: Permite especificar varias pistas de sonido o vídeo para los elementos <AUDIO> y <VIDEO>.
- <METER>: Para trabajar con medidas y escalas.
- <PROGRESS>: Implementa barras de progreso.
- <DATALIST>: Extensión para crear campos con funcionalidad de autocompletar.

- <KEYGEN>: Genera claves pública y privada para encriptación.
- <OUTPUT>: Realizar y mostrar cálculos matemáticos.
- <CANVAS>: Es un elemento que permite la generación de gráficos estáticos ,dinámicos y animaciones. Este objeto puede ser accedido a través de javascript, permitiendo generar gráficos 2D y 3D, animaciones, juegos y composición de imágenes. Actualmente está soportado por la mayoría de los navegadores, incluyendo Internet explorer 9. Con canvas podemos crear rectángulos, líneas, arcos, curvas, dibujar imágenes, añadir colores y estilos, ademas de transformaciones, animaciones y composiciones, permitiéndonos realizar imágenes dinámicas sin la necesidad de utilizar plugins externos.

Elementos para la web semántica:

La segunda clasificación de las nuevas etiquetas de HTML5 están relacionadas con lo que se llama la "web semántica". Se basa en la idea de añadir metadatos semánticos y ontológicos a la web. Esas informaciones adicionales que describen el contenido, el significado y la relación de los datos se deben proporcionar de manera formal, para que así sea posible evaluarlas automáticamente por máquinas de procesamiento. El objetivo es mejorar Internet ampliando la interoperabilidad entre los sistemas informáticos usando agentes inteligentes “programas en las computadoras que buscan información sin operadores humanos”.

Algunas de las nuevas etiquetas nos sirven para decir qué secciones contiene una página.

- <ARTICLE>: Especifica un artículo, es decir, una unidad de contenido.
- <SECTION>: Es una sección dentro de un documento.
- <HEADER:> La cabecera de una página.
- <FOOTER>: El pie de página o informaciones que formen el pie de una sección.
- <ASIDE>: Es una parte de la web que muestra contenido accesorio, generalmente colocado en un panel lateral.
- <NAV>: bloque de navegación del sitio web.

4.2. JQuery Mobile

JQuery es un Framework Javascript, ampliamente usado por muchos desarrolladores de sitios webs. Este Framework sirve para incrementar la velocidad de desarrollo con Javascript, encapsulando muchas tareas comunes que se realizan cuando usamos el lenguaje JavaScript. **[33]**

Por otro lado JQueryMobile consiste en un framework javascript para creación de sitios webs optimizados para los dispositivos móviles más populares. Agrega una capa más al JQuery tradicional y busca suplir algunas necesidades que los programadores de dispositivos móviles padecen.

Antes de que aparecieran estas herramientas, los desarrolladores tenían casi que programar para cada dispositivo en concreto. Esto provocaba muchas horas de trabajo para la creación y mantenimiento de estos sitios.

Con la aparición de JQuery Mobile, nos abstraernos de la lógica específica de cada dispositivo.

Características de JQuery Mobile:

- Facilidad de uso: Otorga mucha facilidad para el desarrollo de interfaces de usuario de dispositivos móviles.

- Soporte HTML5 markup-driven: Cuando utilicemos este framework, nos olvidaremos de tipear código JavaScript. Gracias al uso de etiquetas HTML, que luego en el momento de renderizado serán procesadas por Jquery Mobile.

- Múltiples plataformas soportadas: Los dispositivos móviles han multiplicado el número de navegadores y de plataformas. Tenemos muchos fabricantes, de tablets y smartphones y diversos dispositivos con características distintas, como tamaños de pantallas, sistemas operativos diferentes y diversos navegadores basados en cada uno de ellos. JQuery soporta muchos dispositivos y tecnologías, como ser: IOS, Android, Blackberry, Palm WebOS, Symbian, Windows Mobile, etc.

- Tamaño reducido: Toda la librería comprimida pesa menos de 12K.

- Temas personalizados: El framework expone algunas utilidades para el manejo de temas y también es posible crear temas propios.

- Accesibilidad:Otro punto a favor para este framework es que las aplicaciones generadas a través de él cumplen con los estándares de accesibilidad 1.0. Muchos de los componentes de jQuery Mobile utilizan técnicas como el control del foco, la navegación a través del teclado o los atributos HTML especificados por la especificación W3C WAI-ARIA.

- Preparado para dispositivos táctiles: Los dispositivos táctiles tienen cambios en la gestión de eventos y jQuery Mobile nos facilita la labor de adaptarnos a ellos.

4.3. KineticJS

KineticJS es una librería Javascript que nos permite extender el contexto 2D del tag <CANVAS> de HTML5. Permite animaciones de alto rendimiento, transiciones, anidación de nodos, almacenamiento en caché, filtrado, control de eventos para aplicaciones de escritorio y móviles, etc. **[34]**

Kinetic se compone de 2 capas definidas por el usuario, cada capa tiene 2 contextos. El contexto “escena” es lo que se puede ver, y el contexto buffer es un contexto especial oculto que se utiliza para la detección de eventos de alto rendimiento. Cada capa puede contener formas y grupos de formas. El escenario, capas, grupos y formas son nodos virtuales, similares a los nodos DOM en una página HTML.

Todos los nodos pueden poseer estilos y ser transformados, aunque KineticJS tiene formas predefinidas disponibles, tales como rectángulos, círculos, imágenes, sprites, textos, líneas, polígonos, polígonos regulares, caminos, estrellas, etc, también puede crear formas personalizadas creando una instancia de la clase Shape y la creación de una función de dibujo.

Características:

- API Orientada a Objetos
- Anidación y propagación de eventos
- Alto rendimiento de eventos a través del mapeo en la asignación colores
- Soporte de Capas
- Nodo de almacenamiento en caché para mejorar el rendimiento en los dibujos
- Los nodos pueden ser convertidos en datos de direcciones URL, datos y/o objetos de imagen.
- Soporte de animación y transición
- Drag and drop con restricciones y límites configurables
- Filtros
- Listo para usar formas incluyendo rectángulos, círculos, imágenes, texto, líneas, polígonos, rutas de SVG, etc
- Custom shapes (Formas personalizadas)
- Eventos impulsados por la arquitectura que permite a los desarrolladores suscribirse a eventos, cambios de atributos, etc.
- Serialización y de-serialización
- Soporte de selectores
- Eventos desktop y móviles

4.4. Códigos QR

Un **código QR** (*quick response code*, «código de respuesta rápida») es un módulo para almacenar información en una matriz de puntos o un código de barras bidimensional. Se caracteriza por los tres cuadrados que se encuentran en las esquinas y que permiten detectar la posición del código al lector. La sigla «QR» se deriva de la frase inglesa Quick Response (Respuesta Rápida en español). **[35]**

Figura 10. Figura de un Código QR

Características Generales:

Aunque inicialmente se usó para registrar repuestos en el área de la fabricación de vehículos, hoy los códigos QR se usan para administración de inventarios en una gran variedad de industrias. La inclusión de software que lee códigos QR en teléfonos móviles, ha permitido nuevos usos orientados al consumidor, que se manifiestan en comodidades como el dejar de tener que introducir datos de forma manual en los teléfonos. Las direcciones y los URLs se están volviendo cada vez más comunes en revistas y anuncios . El agregado de códigos QR en tarjetas de presentación también se está haciendo común, simplificando en gran medida la tarea de introducir detalles individuales de un nuevo cliente en la agenda de un teléfono móvil.

Los códigos QR también pueden leerse desde PC, smartphone o tablet mediante dispositivos de captura de imagen, como puede ser un escáner o la cámara de fotos, programas que lean los datos QR y una conexión a Internet para las direcciones web.

Generador de códigos para navegadores web:

Con ciertas extensiones a los navegadores, y generalmente utilizando el menú contextual, que se activa al pulsar el botón derecho del ratón, se puede obtener el código QR del sitio web donde nos encontremos, de un enlace, número de teléfono, SMS, contacto (vcard) o de un texto, lo que hace más fáciles de copiar en un dispositivo móvil.

También podemos generar códigos QR personalizados para una página web en particular.

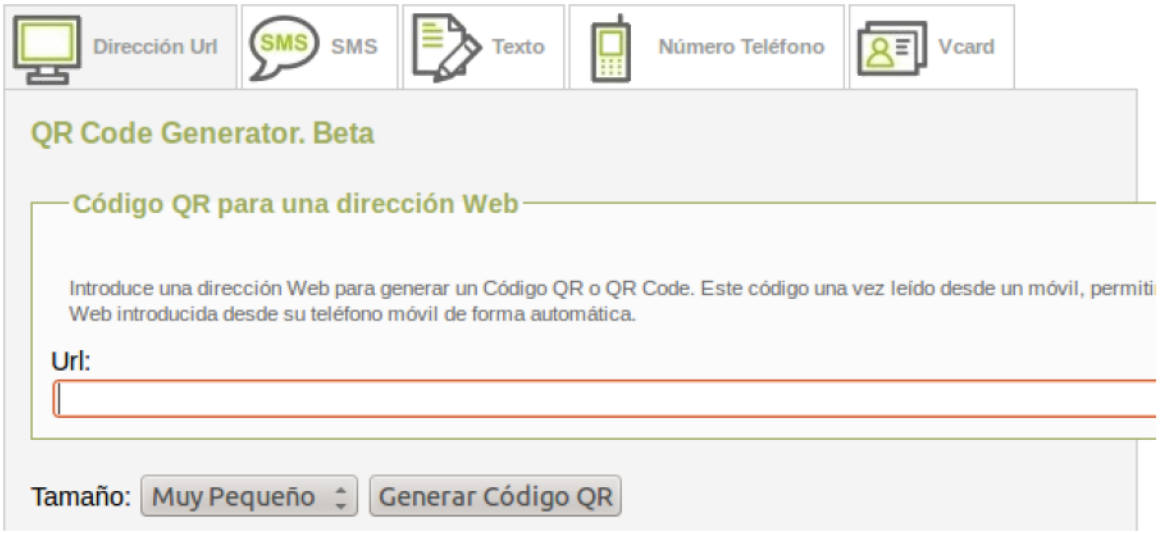


Figura 11. Generador de códigos QR[11]

4.5. WebGL

Antes de introducirnos en las características de WebGL, explicaremos algunos estándares **[36]** necesarios para su utilización

OpenGL (Open Graphics Library) es una especificación estándar que define una API (application programming interface) multiplataforma para escribir aplicaciones que contengan gráficos 2D y 3D. OpenGL esta manejado por el grupo tecnológico Khronos Group, el cual es un consorcio industrial sin fines de lucro encargado de crear estándares abiertos para permitir la creación y aceleración de la computación paralela, gráficos y medios dinámicos en una variedad de plataformas y dispositivos.

OpenGL ES 2.0 (OpenGL for Embedded Systems) es una variante simplificada de OpenGL para dispositivos integrados, tales como smartphones, PDAs, consolas, entre otros. Consiste de un subconjunto bien definido de OpenGL. Permite la programación completa de gráficos 3D. Al igual que OpenGL, esta manejado por Khronos Group. Toda la familia de OpenGL tiene una característica muy importante: la aceleración por hardware, esta consiste en el uso del hardware para desempeñar algunas funciones mucho más rápido de los que es posible en software corriendo en la CPU de propósito general. De esta manera se utiliza la GPU de la tarjeta gráfica para procesar grandes cargas de graficos.

La **GPU** (Graphics Processing unit) o unidad de procesamiento de gráficos es un procesador dedicado al procesamiento de gráficos u operaciones de coma flotante.

WebGL fue creado inicialmente por Mozilla, y más tarde estandarizado por

[11]http://www.codigos-qr.com/generador-de-codigos-qr/

el grupo tecnológico Khronos Group, el mismo grupo responsable de OpenGL y OpenGL ES. El primer prototipo fue diseñado por Mozilla en el año 2006 y a principios del 2009, Mozilla y Khronos Group comenzaron el WebGL Working Group. Además de los ya mencionados, actualmente los principales fabricantes de navegadores, Apple (Safari), Google (Chrome) y Opera (Opera), asi como algunos proveedores de hardware son miembros del grupo de trabajo WebGL o WebGL Working Group. Todos ellos están interesados en verificar que el contenido WebGL pueda correr tanto en desktop y hardware de dispositivos móviles.

WebGL (Web-based Graphics Library) es un estándar web multiplataforma para una API de gráficos 3D de bajo nivel basado en OpenGL ES 2.0 y expuesto a través del elemento canvas de HTML5 como interfaces DOM (Document Object Model). Esta API provee enlaces de JavaScript a funciones OpenGL haciendo posible proveer contenido 3D acelerado en hardware a las páginas web. Esto hace posible la creación de gráficos 3D que se actualizan en tiempo real, corriendo en el navegador.

Podemos decir también que WebGL, es una librería de software que extiende al lenguaje de programación javaScript para permitir generar gráficos interactivos 3D dentro de cualquier navegador web compatible. WebGL es un contexto del elemento canvas que provee un API de gráficos 3D sin la necesidad de plugins. Decir que WebGL es un contexto del elemento canvas, podría no entenderse hasta que se muestre como trabajan juntos en la siguiente sección.

WebGL trae el API de OpenGL ES 2.0 al elemento canvas. El contenido 3D está limitado a canvas. Se hicieron cambios en la API de WebGL en relación con la API de OpenGL ES 2.0 para mejorar la portabilidad a través de varios sistemas operativos y dispositivos.

Podemos observar un modelo conceptual de la arquitectura de WebGL:

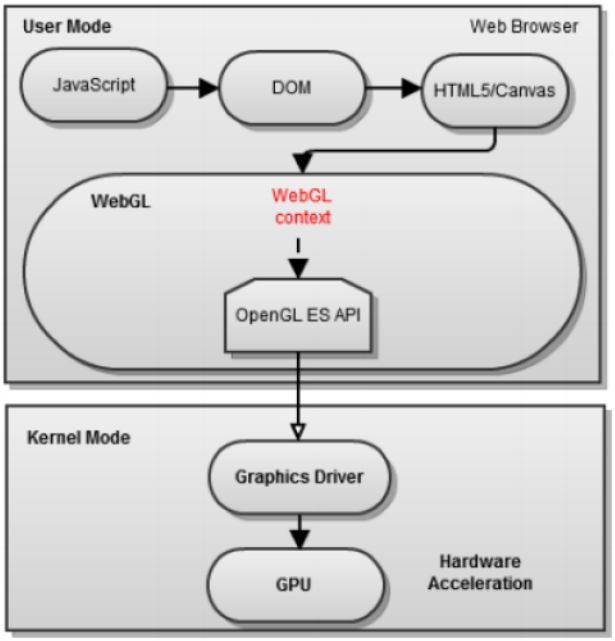


Figura 12. Modelo conceptual de la arquitectura WebGL[12]

[12]http://books.openlibra.com/pdf/WebGL.pdf

En este modelo se utiliza JavaScript para obtener a través del DOM el elemento Canvas de HTML5. Una vez obtenido el elemento Canvas, se define el contexto WebGL, por medio del cual accedemos a la API de WebGL, la cual esta basada en la API de OpenGL ES. Por eso se dice que, técnicamente es un enlace (binding) para JavaScript para usar la implementación nativa de OpenGL ES 2.0. Este último, se encarga de comunicarse con el driver de la tarjeta gráfica y así poder realizar la aceleración por hardware en la GPU. Como vemos, se diferencian los eventos que ocurren en el espacio de usuario y en el kernel.

La API de WebGL interactúa bien con el resto de las plataformas web; específicamente se proporciona apoyo para la carga de texturas 3D a partir de imágenes HTML o video, y la entrada del teclado y mouse se manejan mediante los conocidos eventos DOM. Este API poderoso es de muy bajo nivel y su uso requiere buenos conocimientos acerca de la programación 3D y matemática 3D. Una de las principales desventajas de WebGL es el alto consumo de CPU.

Una característica importante de WebGL es que, brinda la posibilidad de contenidos 3D a la web sin la necesidad de utilizar plugins, ya que se encuentra implementado en el navegador.

WebGL es un estándar web. Un estándar web, es un conjunto de especificaciones técnicas en constante evolución y de buenas prácticas para construir sitios web. Con ello se facilita el mantenimiento, la usabilidad, la interoperabilidad y la calidad de los trabajos. Utilizar un estándar asegura una larga vida a los proyectos, ya que provee por lo menos una pequeña estructura de mantenimiento. También asegura de que la mayoría de las personas puedan visitar el sitio web, sin importar qué navegador se esté utilizando. La compatibilidad hacia adelante y hacia atrás (forward and backward compatibility) es posible.

En el caso particular de WebGL, su API está basado en un estándar de gráficos 3D familiar y ampliamente aceptado. No utilizar un estándar tiene varias desventajas, como poco soporte, mayor exposición a problemas de seguridad y problemas de performance. Además la curva de aprendizaje suele ser muy alta, con un costo-beneficio menor. Otro problema que se puede dar es el del vendor lock in, en donde se tiene una dependencia absoluta del proveedor, el cual puede tener un costo elevado.

El lanzamiento (release) de la especificación de la versión nal de WebGL 1.0 se dio el 3 de marzo del 2011 en la conferencia de desarrolladores de juego (Game Developers Conference - GDC), en San Francisco, Estados Unidos

4.5.1 Seguridad en WebGL

Al decir que WebGL da a las páginas Web acceso al hardware gráfico, varias personas se preocupan de que WebGL sea una grave amenaza de seguridad. En mayo del año 2011 la firma de seguridad Context Information Security descubrió una vulnerabilidad en WebGL que podría permitir a los atacantes ejecutar código malicioso de forma remota y tomar el control de los equipos.

Se abre una línea directa desde Internet a la GPU del sistema, abriendo un gran hueco de seguridad. Entonces, con las versiones actuales de WebGL, un hacker podría escribir una página WebGL que haga que la tarjeta gráfica deje de responder a otras aplicaciones.

Con respecto al problema, Google dijo que muchos procesos WebGL, incluyendo el de la GPU, se ejecutan por separado y dentro del Sandbox de Chrome (filtro) como medida para prevenir posibles ataques.

El grupo Mozilla arma que han tomado sus precauciones para evitar al máximo que esto no suceda, por lo que cuentan con una lista de drivers admitidos.

También lo hizo Khronos Group, diciendo que se están evaluando estas advertencias y que los fabricantes de la GPU están añadiendo soporte para un mecanismo que ayudara a resolver el problema.

Microsoft ha anunciado que no soportará el estándar WebGL por considerar que es una fuente continua de vulnerabilidades difíciles de corregir. En un blog reciente, Microsoft dijo: “En su forma actual, WebGL no es una tecnología que Microsoft pueda apoyar desde una perspectiva de seguridad". Microsoft llega a la conclusión de que WebGL “tendrá dificultades pasando los requerimientos del ciclo de vida para el desarrollo de la seguridad de Microsoft". Según Microsoft, además del problema de seguridad mencionado en la sección anterior, WebGL puede ser vulnerable a ataques DoS (Denial of Service) y señalan que su seguridad dependen de los niveles más bajos del sistema, incluyendo los drivers de OEM.

Esta postura de Microsoft, impide que WebGL se convierta en un estándar soportado por todos los principales navegadores. El soporte universal significa que todos los desarrolladores web podrán contar con WebGL estando disponible y por lo tanto usarlo. Su ausencia significa que algunos sitios y aplicaciones web requerirán chequeos de compatibilidad. Para algunos, los problemas de seguridad de WebGL, es una excusa, ya que si se expande WebGL, tanto DirectX como Silverlight saldrán perjudicadas.

Como funciona el ataque:

El mecanismo del ataque se pueden resumir en los siguientes 4 pasos:

- Un usuario visita un sitio web donde un código WebGL malicioso se encuentra alojado.
- El componente WebGL carga las geometrías 3D específicas y el código a la tarjeta gráfica.
- La geometría o el código malicioso explota los errores en los drivers de la tarjeta gráfica.
- El hardware gráfico puede ser atacado causando que todo el sistema se congele o caiga.

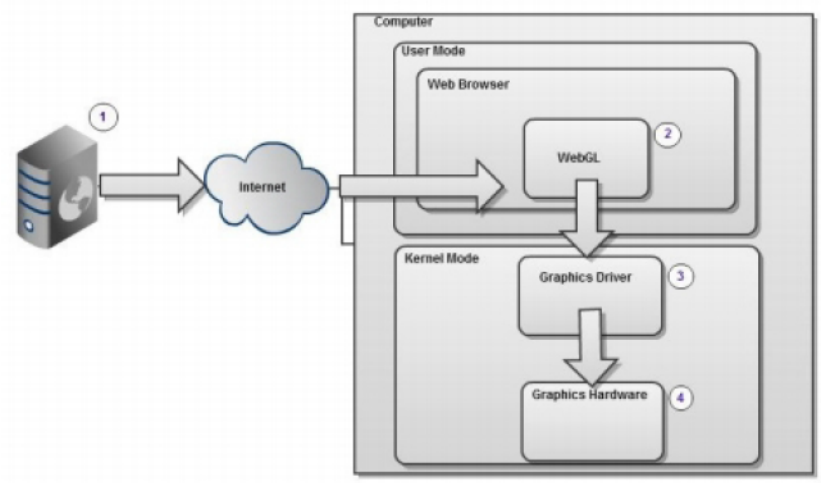


Figura 13. El mecanismo de un ataque WebGL[13]

4.5.2 Contexto 3D : WebGL

Para trabajar con el canvas 3D existen dos métodos: uno de ellos implica desarrollar un plano 2D desde un objeto 3D, controlando el plano y la cámara que queremos mostrar. La otra alternativa, y la que nos interesa es utilizando WebGL.

Como ya vimos en la implementación de la interfaz del prototipo, para trabajar en 2D, debemos utilizar el contexto “2d" del elemento canvas, y para trabajar en WebGL debemos utilizar el contexto “webgl" o en caso de que no funcione, el “experimental-webgl". Las cadenas son case sensitive. Una vez obtenido el contexto para WebGL utilizando el elemento canvas, podemos empezar a trabajar con WebGL.

Antes de ver cómo implementamos la interfaz 3D del prototipo veremos una serie de funciones que nos ayudarán a entender mejor cómo funciona WebGL y que facilitarán la explicación de la implementación del prototipo. Estas funciones deben ir dentro de la etiqueta <*script*> ,ya que se encuentran escritas en javaScript.

Comenzamos definiendo el tag de HTML 5 <canvas>

```
<canvas id="micanvas" width="200" height="150">
    El navegador no soporta el elemento canvas de HTML5.
</canvas>
```

Para empezar, se muestra una función que recibe como parámetro el elemento canvas obtenido a través del DOM, y que se encarga de inicializar el contexto WebGL.

```
function initWebGL(canvas){
      contexto = canvas.getContext("webgl");
      if (!contexto) {
            contexto = canvas.getContext("experimental-webgl");
      }
```

[13]http://books.openlibra.com/pdf/WebGL.pdf

```
      if (!contexto) {
            alert("Tu navegador no soporta WebGL");
            return;
      }
}
```

`getContext()` nos devuelve el contexto, Una práctica buena es comprobar si se soporta el contexto canvas, ya que si el navegador no lo soporta muestre un mensaje entendible y no un error.

Ya analizada la función anterior, se procede a crear la funcion start() que será llamada en el body, de la siguiente manera:

```
<body onload="start()">
```

Esta función obtendrá el elemento canvas, y llamara a la función initWebGL() para obtener el contexto WebGL.

```
function start(){
      var canvas = document.getElementById("micanvas");
      initWebGL(canvas);
      if (contexto){
            //Nuestro codigo WebGL
      }
}
```

Desde ahora podemos empezar a crear nuestros fragmentos de código para probar WebGL.

En el siguiente código sencillo WebGL se muestra información acerca de la versión de WebGL, el navegador, y el renderizador usado. Como vemos, se utilizan algunos atributos y métodos del objeto contexto.

```
alert(
      "WebGL version=" + contexto.getParameter(contexto.VERSION) +"\n"+
      "WebGL vendor=" + contexto.getParameter(contexto.VENDOR) +"\n"+
      "WebGL renderer="+ contexto.getParameter(contexto.RENDERER)+"\n"
);
```

Hasta ahora, sólo se realizó una inicialización del contexto WebGL, pero para trabajar con WebGL se requieren de muchas cosas mas. Una vez que se ha obtenido el contexto WebGL, se debe crear el buffer de dibujo (drawing buffer) en el cual son renderizadas todas las llamadas a la API. El tamaño del buffer,está determinado por los atributos width y height del elemento canvas. Se debe limpiar (clear) el buffer con un color y una profundidad (Depth) específica, cuyos valores por defecto son (0,0,0,0) y 1.0 respectivamente. Luego, se deben crear unos buffers con los cuales trabajar.

Podemos expandir nuestra función start(), agregando dos funciones, realizando el clear del buffer de dibujo mencionado, y una llamada a la función drawScene().

```
function start(){
      var canvas = document.getElementById("micanvas");
      initWebGL(canvas);
      initShaders();
      initBuffers();
      contexto.clearColor(0.0,0.0,0.0,1.0);
      contexto.enable(contexto.DEPTH_TEST);
      drawScene();
}
```

En initBuffers(), se crean los buffers que almacenarán los vértices de los gráficos. Los vértices son los puntos en el espacio 3D que definen la figura que estamos dibujando. Este buffer se crea con el método del objeto contexto, contexto.createBuffer(). Este buffer, es en realidad una sección de memoria en la tarjeta gráfica.

Para manejar los buffers, existen dos métodos, contexto.bindBuffer() y contexto.bufferData(). El primero especifica el buffer actual, en el cual deben realizarse las operaciones y el segundo permite cargar el buffer actual. La función drawScene(), es el lugar en donde utilizamos los buffers para dibujar la imagen que se ve en pantalla. En ella se utiliza el método viewport() el cual define el lugar de los resultados de renderización en el buffer de dibujo.

Sin el viewport, no se manejará adecuadamente el caso donde canvas cambie de tamaño. Luego, se define la perspectiva con el cual queremos observar la escena. Por defecto, WebGL dibuja las cosas que se encuentran cerca con el mismo tamaño que el de las cosas que se encuentran alejadas. Para definir que las cosas alejadas se vean más pequeñas, hay que modificar la perspectiva. Para empezar a dibujar, hay que moverse al centro de la escena 3D. Al dibujar una escena, se especifica la posición actual y la rotación actual, ambos mantenidos en una matriz. La matriz que se usa para representar el estado actual move/rotate es llamado model-view matrix. Esta matriz nos mueve al punto de
origen desde el cual podemos empezar a dibujar.

La funcion initShaders(), inicializa los Shaders, evidentemente. Un Shader es un conjunto de instrucciones de software que es utilizado para realizar transformaciones y crear efectos especiales, como por ejemplo iluminación, fuego o niebla. Proporciona una interacción con la GPU hasta ahora imposible, por lo cual la renderización se realiza en hardware gráfico a gran velocidad. Se utilizan los shaders de manera que los segmentos de programa gráficos sean ejecutados en la GPU, y no en javaScript, lo cual sería muy ineficiente y lento.

Un lenguaje de sombreado muy utilizado es GLSL (OpenGL Shading Language), el cual es un lenguaje de alto nivel basado en C. Por eso, algunos desarrolladores reconocen a WebGL como una API basado en el uso de shaders GLSL.

Como vimos, utilizar WebGL tiene su complejidad inherente, debido a su API de bajo nivel. OpenGL maneja muchos tipos de recursos como parte de su estado. WebGL representa estos recursos como objetos DOM. Los recursos que son soportados actualmente son: texturas, buffers, framebuffers, renderbuffers, shaders y programas. La interfaz del contexto tiene un método para crear un objeto para cada tipo.

Es importante mencionar que podemos agregar contenido 2D al contexto WebGL, pero debemos tener en cuenta que se está dibujando sobre un espacio tridimensional.

4.6. Symfony

Symfony es un framework PHP provisto de un gran conjunto de herramientas y tecnologías de desarrollo. **[37]**

En Symfony basta con definir un modelo de datos en formato YAML (schema.yml) para poder generar mediante un comando de consola un modelo de objetos por cada una de las entidades expuestas. Así mismo, es posible crear módulos para los objetos, donde cada módulo tiene operaciones predefinidas (borrar, crear, editar, listar, etc). También por cada módulo creado, se genera una Clase Form (Que permite manipular la carga/edición y/o creación de un objeto) y una clase Filter (Que permite manipular por cada uno de los atributos del objeto las opciones de filtrado en listados).

Symfony está basado en el Patrón de Diseño MVC - **M**odel-**V**iew-**C**ontroller (Modelo - Vista - Controlador). La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

Otra de las bondades de symfony es que cuenta con una gran comunidad que desarrolla plugins. Los plugins permiten agrupar todo el código diseminado por diferentes archivos y reutilizar este código en otros proyectos. Los plugins permiten encapsular clases, filtros, *mixins*, *helpers*, archivos de configuración, tareas, módulos, esquemas y extensiones para el modelo, *fixtures*, archivos estáticos, etc. Los plugins son fáciles de instalar, de actualizar y de desinstalar. Se pueden distribuir en forma de archivo comprimido .tgz, un paquete PEAR o directamente desde el repositorio de código. La ventaja de los paquetes PEAR es que pueden controlar las dependencias, lo que simplifica su actualización. La forma en la que

Symfony carga los plugins permite que los proyectos puedan utilizarlos como si fueran parte del propio framework.

En otras palabras, un plugin es una extensión encapsulada para un proyecto Symfony. Los plugins permiten no solamente reutilizar código propio, sino que permiten aprovechar los desarrollos realizados por otros programadores y permiten añadir al núcleo de Symfony extensiones realizadas por otros desarrolladores.

Por ejemplo, un plugin muy utilizado en symfony es sfGuardDoctrinePlugin, el cual permite incluir autenticación, autorización y otras opciones de gestión de usuarios avanzadas (como grupos y credenciales).

5.Características a contemplar en la Navegación Indoor

Detallaremos la funcionalidad que provee el sistema para cumplir con su objetivo.

Casos de Uso

Usuarios Registrados

1) Crear Estructura

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado da de alta una estructura en el sistema
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se crea una nueva estructura en el sistema

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, crear una estructura.
1.4 El usuario registrado selecciona del menú la función, crear una estructura.	
1.5 El usuario registrado ingresa los atributos: Nombre, Tipo de Estructura y Capacidad.	
	1.6 El sistema valida la información enviada por el usuario y crea la nueva estructura.

2) Editar Estructura

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado selecciona una estructura para ser editada
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se cambian los atributos de una estructura

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA

1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, editar una estructura.
1.4 El usuario registrado busca en el listado de estructuras la que desea modificar y selecciona la opción de editar.	
1.5 El usuario registrado modifica los atributos necesarios.	
	1.6 El sistema valida la información enviada por el usuario y modifica los atributos de la estructura.

3) Borrar Estructura

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado selecciona una estructura para ser eliminada
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se elimina la estructura del sistema

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, borrar una estructura.
1.4 El usuario registrado busca en el listado de estructuras la que desea eliminar y selecciona la opción borrar.	
	1.5 El sistema pregunta al usuario si realmente desea realizar el borrado físico de la estructura.
1.6 El usuario registrado confirma que desea eliminar la estructura.	
	1.7 La estructura es eliminada del sistema.

4) Crear Punto

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado da de alta un punto en el sistema
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema

POSTCONDICIÓN: Se crea un nuevo punto en el sistema

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado registrado. Entre ellas, crear un punto.
1.4 El usuario registrado selecciona del menú la función, crear un punto.	
1.5 El usuario registrado ingresa los atributos: Punto de origen X y Punto de Origen Y. También puede seleccionar una estructura para ser asociada con el punto.	
	1.7 El sistema valida la información enviada por el usuario y crea un nuevo punto en el sistema

5) Editar Punto

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado selecciona un punto para ser editado
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se cambian los atributos de un punto

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, editar un punto.
1.4 El usuario registrado busca en el listado de puntos el que desea modificar y selecciona la opción de editar.	
1.5 El usuario registrado modifica los atributos necesarios.	
	1.6 El sistema valida la información enviada por el usuario y modifica los atributos del

	punto.

6) Borrar Punto

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado selecciona un punto para ser eliminado
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se elimina el punto del sistema

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, borrar un punto.
1.4 El usuario registrado busca en el listado de puntos el que desea eliminar y selecciona la opción borrar.	
	1.5 El sistema pregunta al usuario si realmente desea realizar el borrado físico del punto.
1.6 El usuario registrado confirma que desea eliminar el punto.	
	1.7 El punto es eliminado del sistema.

7) Crear Punto de Navegación

PARTICIPANTES: Usuario Registrado, Sistema
DESCRIPCIÓN: El Usuario Registrado da de alta un punto de navegación en el sistema
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se crea un nuevo punto de navegación en el sistema

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas,

	crear un punto de navegación.
1.4 El usuario registrado selecciona del menú la función crear punto de navegación.	
1.5 El usuario registrado ingresa los atributos: Nombre y selecciona el punto con el que el punto de navegación está asociado.	
	1.7 El sistema valida la información enviada por el usuario y crea el nuevo punto de navegación.

8) Editar Punto de Navegación

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado selecciona un punto de navegación para ser editado
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se cambian los atributos de un punto de navegación

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, editar un punto de navegación.
1.4 El usuario registrado busca en el listado de puntos de navegación el que desea modificar y selecciona la opción de editar.	
1.5 El usuario registrado modifica los atributos necesarios.	
	1.6 El sistema valida la información enviada por el usuario y modifica los atributos del punto de navegación.

9) Borrar Punto de Navegación

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado selecciona un punto de navegación para ser eliminado
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se elimina el punto de navegación del sistema

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, borrar un punto de navegación.
1.4 El usuario registrado busca en el listado de puntos de navegación el que desea eliminar y selecciona la opción borrar.	
	1.5 El sistema pregunta al usuario si realmente desea realizar el borrado físico del punto e navegación.
1.6 El usuario registrado confirma que desea eliminar el punto de navegación.	
	1.7 El punto de navegación es eliminado del sistema.

10) Crear Relación entre Puntos de Navegación

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado da de alta una relación entre dos puntos de navegación.
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se crea una nueva relación entre dos puntos de navegación.

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, crear una relación entre puntos de navegación.
1.4 El usuario registrado selecciona del menú la función, crear un relación entre puntos de navegación	
1.5 El usuario registrado ingresa los atributos: Punto de Navegación 1, Puntos de Navegación 2 y la distancia que hay entre los mismos.	
	1.6 El sistema valida la información enviada por el usuario y crea la relación entre los puntos de navegación seleccionados.

11) Editar Relación entre Puntos de Navegación

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado selecciona una relación entre puntos de navegación para ser editada
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se modifican los atributos de una relación entre dos puntos de navegación

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del usuario.
	1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, editar una relación entre puntos de navegación.
1.4 El usuario registrado busca en el listado de relaciones entre puntos de navegación la que desea modificar y selecciona la opción de editar.	
1.5 El usuario registrado modifica los atributos necesarios.	
	1.6 El sistema valida la información enviada por el usuario y modifica los atributos de la relación entre puntos de navegación.

12) Borrar Relación entre Puntos de Navegación

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado selecciona una relación entre puntos de navegación para ser eliminada
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se elimina la relación entre puntos de navegación del sistema

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	
	1.2 El sistema valida la información del

	usuario. 1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, borrar una relación entre puntos de navegación.
1.4 El usuario registrado busca en el listado de relaciones entre puntos de navegación y elige la que desea eliminar seleccionando la opción borrar.	1.5 El sistema pregunta al usuario si realmente desea realizar el borrado físico de la relación entre puntos de navegación.
1.6 El usuario registrado confirma que desea eliminar la relación entre puntos de navegación.	1.7 La relación entre puntos de navegación es eliminada del sistema.

13) Crear Multimedia

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado da de alta un recurso multimedia en el sistema
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se crea una nuevo recurso multimedia en el sistema

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	1.2 El sistema valida la información del usuario. 1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, crear multimedia.
1.4 El usuario registrado selecciona del menú la función, crear multimedia. 1.5 El usuario registrado ingresa los atributos: Nombre, tipo de multimedia y selecciona la estructura con la cuál el recurso multimedia estará asociado.	1.6 El sistema valida la información enviada por el usuario y crea un recurso multimedia en el sistema

14) Editar Multimedia

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado selecciona un recurso multimedia para ser editado
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se cambian los atributos de un recurso multimedia

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	1.2 El sistema valida la información del usuario. 1.3 El sistema despliega las funciones disponibles al usuario registrado. Entre ellas, editar un recurso multimedia.
1.4 El usuario registrado busca en el listado de recursos multimediales el que desea modificar y selecciona la opción de editar. 1.5 El usuario registrado modifica los atributos necesarios.	1.6 El sistema valida la información enviada por el usuario y modifica los atributos del recurso multimedia.

15) Borrar Multimedia

PARTICIPANTES: Usuario registrado, Sistema
DESCRIPCIÓN: El Usuario registrado selecciona un recurso multimedia para ser eliminado
PRECONDICIÓN: El usuario debe encontrarse logueado en el sistema
POSTCONDICIÓN: Se elimina el recurso multimedia del sistema

CURSO DE EVENTOS:

USUARIO REGISTRADO	SISTEMA
1.1 Un usuario realiza inicio de sesión utilizando su nombre de usuario y contraseña.	1.2 El sistema valida la información del usuario. 1.3 El sistema despliega las funciones disponibles al usuario. Entre ellas, borrar un recurso multimedia.

1.4 El usuario registrado busca en el listado de recursos multimediales el que desea eliminar y selecciona la opción borrar.	1.5 El sistema pregunta al usuario si realmente desea realizar el borrado físico del recurso multimedia.
1.6 El usuario registrado confirma que desea eliminar el recurso multimedia.	1.7 El recurso multimedia es eliminado del sistema.

Usuarios Comunes

1) Leer Posición Actual

PARTICIPANTES: Usuario, Sistema
DESCRIPCIÓN: El usuario lee un código QR desde su posición actual
PRECONDICIÓN: Debe existir un código QR en el lugar donde se encuentra el usuario
POSTCONDICIÓN: El usuario obtiene una lista de estructuras a donde navegar

CURSO DE EVENTOS:

USUARIO	SISTEMA
1.1 Desde su posición actual, el usuario lee un código QR.	1.2 El sistema recibe la información de la posición actual del usuario y la guarda en una variable. 1.3 El sistema presenta al usuario un buscador de estructuras y un listado con todos las estructuras posibles navegables.

2) Buscar Destino

PARTICIPANTES: Usuario, Sistema
DESCRIPCIÓN: El usuario busca un destino (estructura) a donde desea navegar
PRECONDICIÓN: El usuario tiene que haber leído un código QR
POSTCONDICIÓN: El usuario obtiene la estructura a la cual desea dirigirse

CURSO DE EVENTOS:

USUARIO	SISTEMA
1.2 El usuario ingresa parte del nombre de la estructura a la cuál desea navegar.	1.1 El sistema presenta al usuario un buscador de estructuras y una lista con todas las estructuras disponibles y navegables. 1.3 El sistema acota el listado de estructuras basándose en el nombre ingresado por el

	usuario.

3) Navegar a Destino

PARTICIPANTES: Usuario, Sistema
DESCRIPCIÓN: El usuario navega desde el origen al destino
PRECONDICIÓN: El usuario filtro y encontró el destino al que desea navegar
POSTCONDICIÓN: El usuario obtiene un mapa con una ruta desde su posición actual hasta el lugar de destino.

CURSO DE EVENTOS:

USUARIO	SISTEMA
3.1 El usuario selecciona el destino al que desea navegar.	
	3.2 El sistema procesa la información enviada por el usuario realizando el cálculo del camino mínimo entre el punto de origen actual y el destino.
3.3 El usuario visualiza en su interfaz un plano con la un camino desde su posición actual y hasta el destino seleccionado.	

4) Ver detalle de Estructura

PARTICIPANTES: Usuario, Sistema
DESCRIPCIÓN: El usuario selecciona una estructura del mapa para visualizar en detalle
PRECONDICIÓN: El usuario tiene que estar navegando a un destino
POSTCONDICIÓN: El usuario obtiene un detalle (recursos multimediales, nombre, capacidad, etc) de la estructura seleccionada.

CURSO DE EVENTOS:

USUARIO	SISTEMA
4.1 El usuario selecciona una estructura del plano.	
	4.2 El sistema procesa la petición del usuario recibiendo el identificador de la estructura seleccionada.
	4.3 El sistema busca toda la información asociada con la estructura.
4.4 El usuario visualiza en su interfaz toda la información de la estructura seleccionada.	

5) Modificar Destino

PARTICIPANTES: Usuario, Sistema
DESCRIPCIÓN: El usuario modifica su destino actual

PRECONDICIÓN: El usuario debe de encontrarse en el detalle (ver) de una estructura
POSTCONDICIÓN: El usuario obtiene un camino desde su ubicación actual hasta el nuevo lugar seleccionado

CURSO DE EVENTOS:

USUARIO	SISTEMA
5.1 El usuario selecciona desde el detalle de una estructura que desea dirigirse a la misma.	
	5.2 El sistema procesa la información enviada por el usuario realizando el cálculo del camino mínimo entre el punto de origen actual y el nuevo destino seleccionado.
5.3 El usuario visualiza en su interfaz un plano con un camino desde su posición actual hasta el nuevo destino seleccionado.	

6.Modelo para Navegación Indoor

6.1.Modelo de clases

El modelo de clases permite representar toda la información necesaria para la representación del edificio en donde se navegará y los puntos navegables.

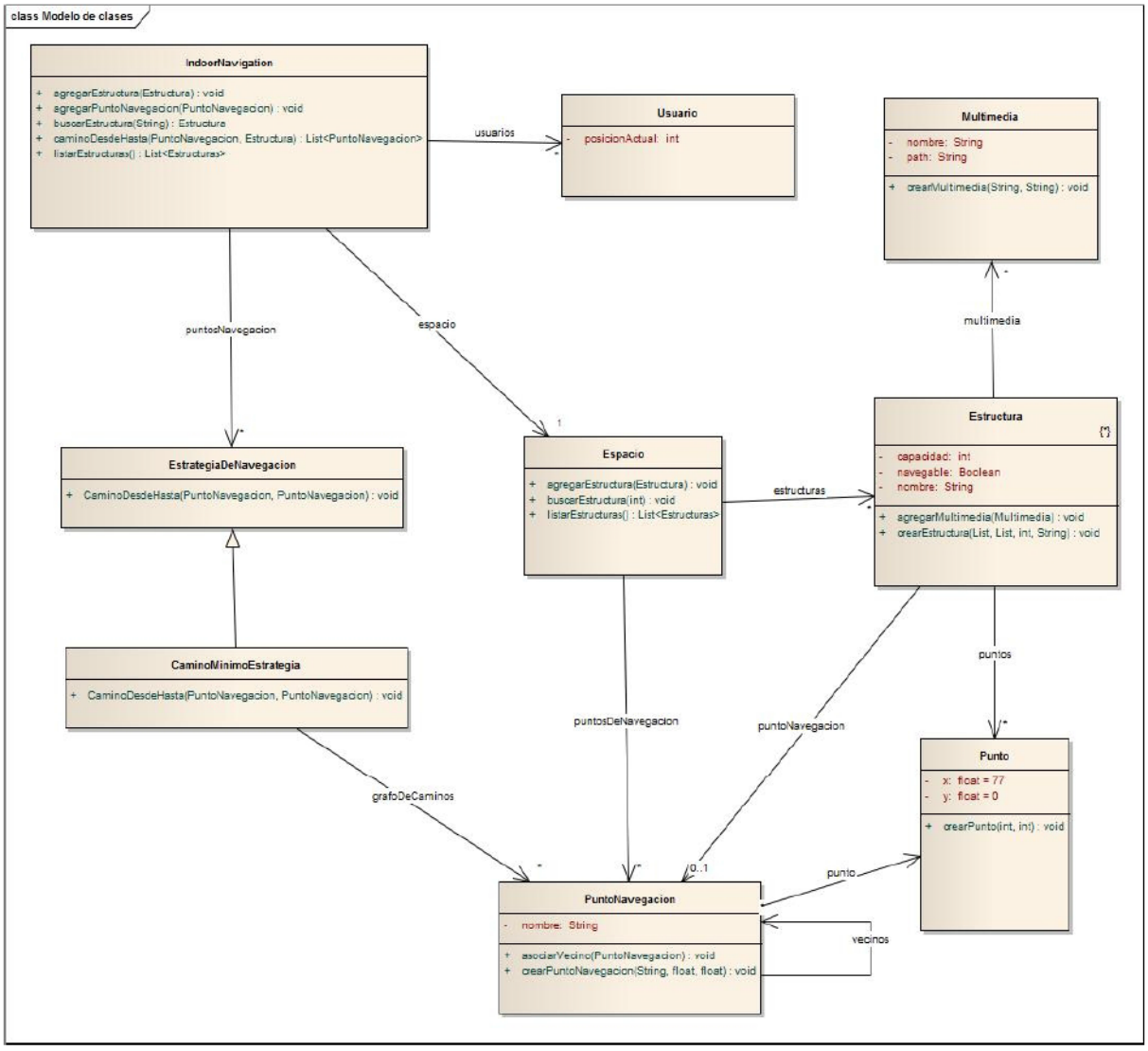


Figura 14. Modelo de clases de la aplicación IndoorNavigation

Detallaremos cada una de las clases definidas:

- **IndoorNavigation**: Representa a la aplicación en sí, es la encargada de responder a las peticiones de los usuarios. Conoce a un Espacio, el cual representa el edificio, sus estructuras y caminos posibles dentro de él. Tiene

también una Estrategia de Navegación, la cual determinará el algoritmo a utilizar para el cálculo de caminos.

- **Usuario**: representa al usuario del sistema, el cual mediante peticiones al sistema podrá determinar su ubicación actual, ver los posibles destinos dentro del edificio, y una vez seleccionado un destino, será guiado hasta este por la aplicación.
- **Espacio**: Es la abstracción del edificio navegable. Contiene puntos de navegación y estructuras. Permite agregar nuevas estructuras, listarlas y obtener el grafo de caminos.
- **Estructura**: Representa a cada sala o habitación dentro de un edificio. La forma y ubicación de una Estructura esta dada por los puntos que la componen, tendrá un Punto por cada vértice de la estructura que deseemos representar, permitiendonos asi representar cualquier forma. Una estructura puede ser navegable o no, si es navegable tiene un punto de navegación asociada a la misma para que pueda ser seleccionada como destino u origen de la navegación. Una estructura también tiene datos asociados a la misma, como fotos, videos y texto, que ayudarán al usuario a crear una relación entre la estructura dibujada en un plano y la estructura real.
- **Punto**: están formados por un par (x,y) que ubica al punto en un plano. Esto será utilizado para poder representar un vértice de estructuras, o la posición de un punto de navegación.
- **PuntoNavegacion**: Representan puntos por donde un usuario puede moverse y ser posicionado. Algunos puntos de navegación están asociados a estructuras convirtiéndolos en posibles destinos. Los puntos de navegación se relacionan entre sí, guardando como dato de esa relación la distancia a ese punto de navegación, formando así una red o grafo, esto es lo que determina los caminos dentro del edificio y hace posible el cálculo del camino de un punto de navegacion de origen a uno de destino.
- **Multimedia**: Representan información visual asociada a una estructura, de esta manera el usuario no solo verá indicaciones de ruteo al destino sino que podrá ver información relacionada al lugar donde se dirige o está parado.
- **EstrategiaDeNavegacion**: Es una jerarquía de algoritmos de cálculos de caminos. Permite tener varias estrategias concretas de cálculo de camino instanciadas y cambiar de estrategia en tiempo de ejecución. De esta forma podemos tener diferentes criterios para determinar un camino.
- **CaminoMinimoEstrategia**: Estrategia de cálculo de caminos concreta, en este caso esta estrategia determinará el camino de menor distancia de un punto de navegacion a otro. Esta es la estrategia que implementaremos en la aplicación.

6.2.Interacción y diagramas de secuencia.

A continuación se representa en diagramas de secuencia los 2 casos de uso más importantes en el sistema: Selección de destino y Navegar a destino.

Selección de destino.

En el siguiente diagrama de secuencia podemos ver como el usuario interactúa con el sistema para poder seleccionar el destino al cual desea dirigirse.

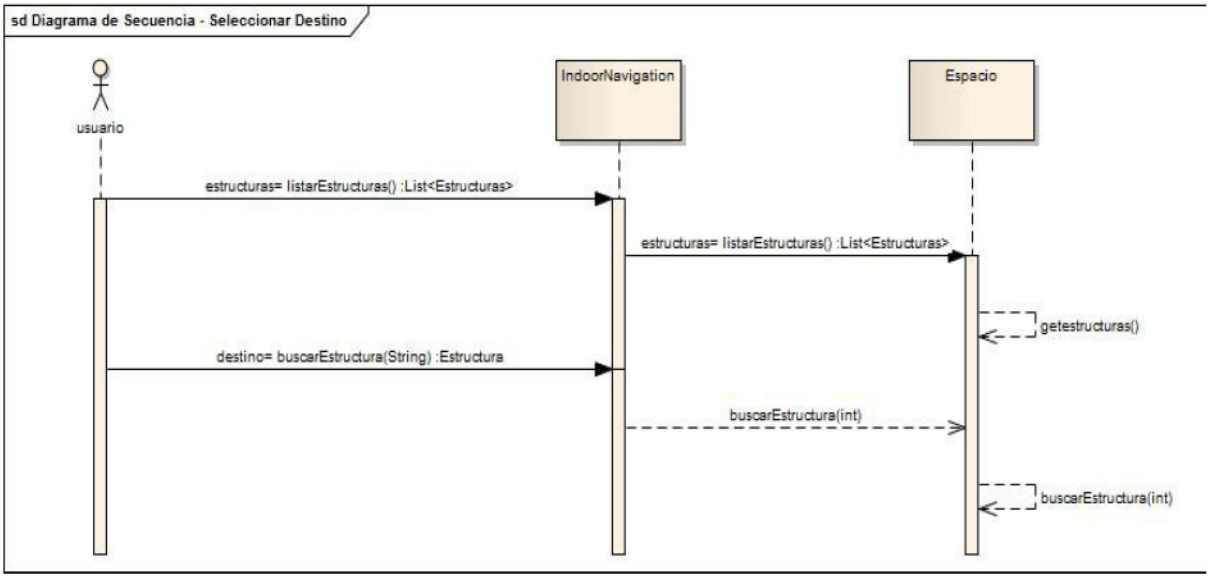


Figura 15. Diagrama de secuencia Selección de destino

La interacción comienza cuando el usuario ingresa al sistema leyendo un código QR, el cual contiene la url del sistema y un parámetro indicando la posición actual, el cual será el punto de origen.
El sistema responde listando todas las estructuras navegables para que el usuario pueda elegir su destino.
El usuario elige su destino de la lista de estructuras presentadas por el sistema.

Navegar a Destino

En este punto el usuario ya tiene asociado un punto de navegacion de origen y uno de destino. El siguiente paso es indicarle al sistema que realice el cálculo de camino mínimo entre estos dos puntos de navegación y que éste sea presentado al usuario en la interfaz de la aplicación.

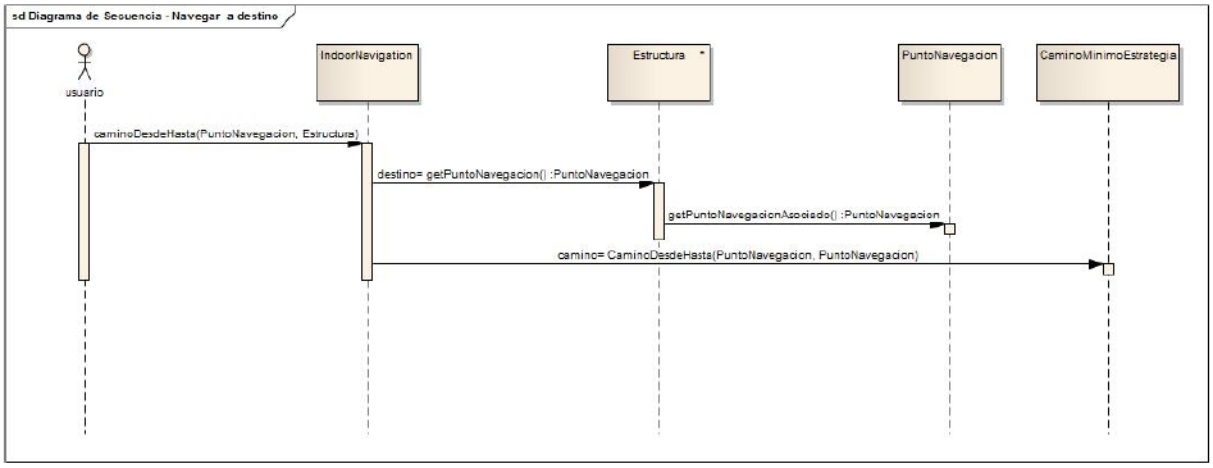


Figura 16. Diagrama de secuencia Navegar a destino

El usuario inicia la interacción pidiendo al sistema el cálculo de caminos de un origen a un destino.
IndoorNavigation busca la estructura referenciada como destino y recupera el punto de navegación asociado a esta.
Una vez obtenido el punto de navegacion de origen y destino se delega a la estrategia de cálculos de camino para que esta determine el camino, el cual será representado en un plano en la interfaz del sistema.

Ver detalles de Estructura

Cuando el usuario está visualizando el mapa del edificio puede consultar más información sobre una estructura.

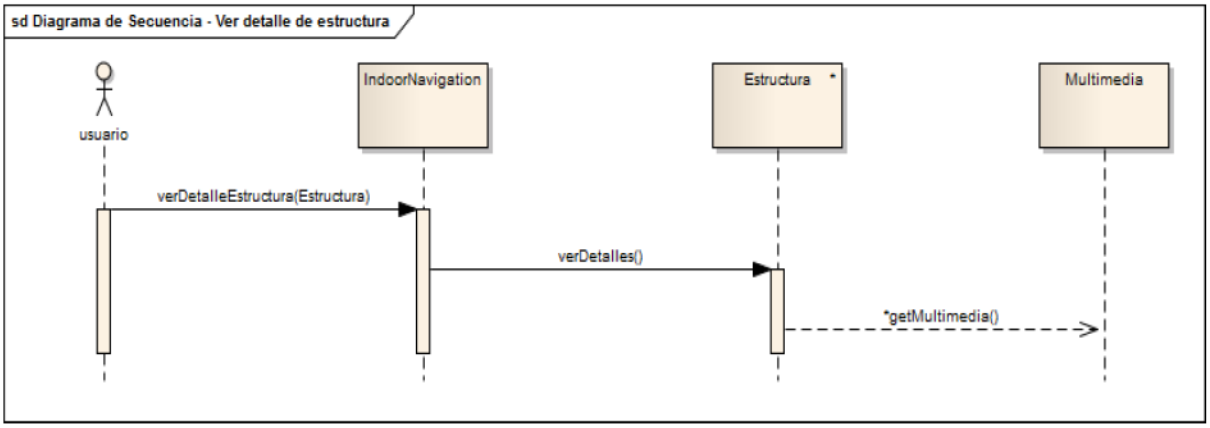


Figura 17. Diagrama de secuencia Ver detalle de estructura

El usuario selecciona una estructura y elige la opción “ver detalles de la estructura”.
IndoorNavigation recibe la estructura y le pide a ésta los recursos multimedia que tiene asociados para que lo mismos sean representados en la interfaz.

6.3.Ejemplo Instanciación

La Instanciación del edificio navegable es uno de los puntos más importantes de la aplicación, ya que es donde se definen las estructuras que actuarán como posibles destinos del usuario y el grafo que formarán los puntos de navegación. La instanciación es un paso complejo ya que se crean partes del edificio navegable individualmente para luego ser asociadas, por este motivo para ver el edificio resultante necesitamos tener todas las estructuras con sus puntos creados y asociadas a sus respectivos puntos de navegación.
Para poder ejemplificar la instanciación, tomaremos un plano de un edificio simplificado, el cual cuenta con tres Estructuras navegables elegibles como destino.

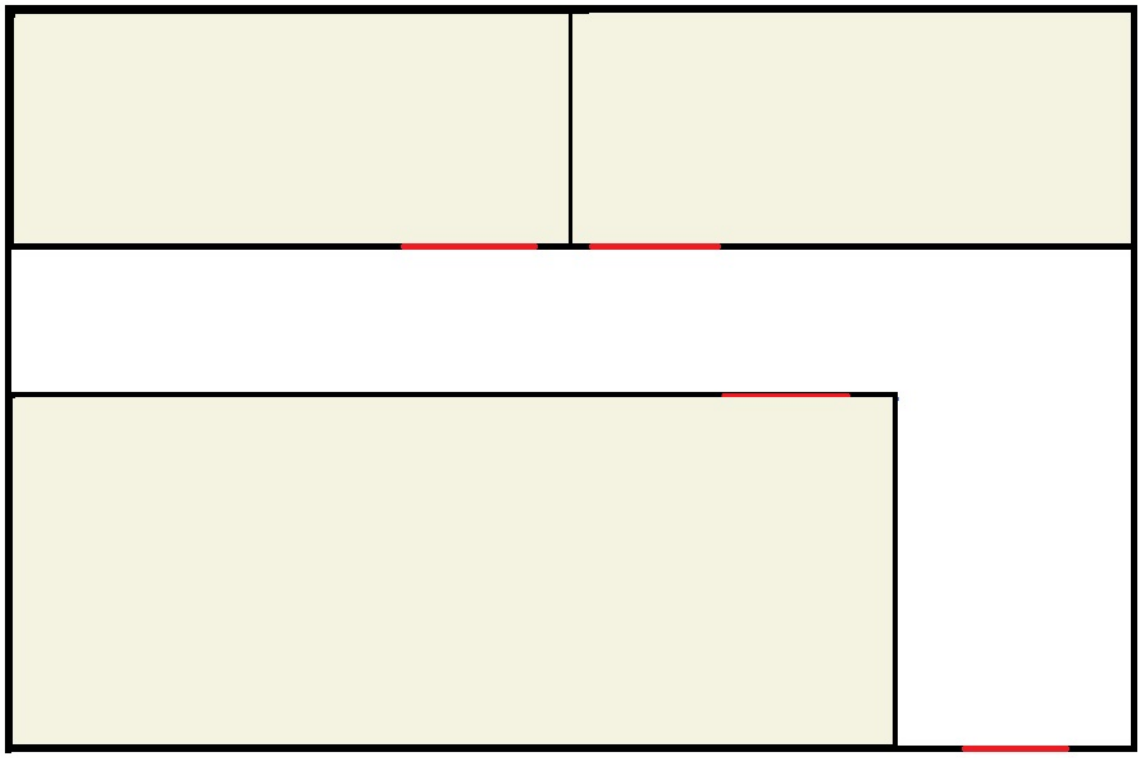


Figura 18. Ejemplo edificio simplificado

Para poder representar este plano con el modelo de clases elegido es necesario definir más elementos que nos permiten instanciar el edificio:

- **Puntos**: los puntos representan los vértices de cada estructura (sea navegable o no) y la posición de los puntos de navegación. A cada uno de los puntos definidos en el plano le fué asignado una coordenada (x,y) que la aplicación luego usará para determinar su posición en el plano.
- **Puntos de navegación**: los puntos de navegación determinan los caminos posibles dentro del edificio. Estos también tienen un punto asociado para

situarlos en el plano. Cada punto de navegación está asociado con los puntos de navegación cercanos y esta relación guarda como información la distancia entre estos, creando así un grafo que luego la aplicación utilizará para determinar el camino. Se pueden ver que las tres estructuras tienen un punto de navegación asociado, esto las convierte en navegables, y de esta manera, elegibles como destinos al usuario de la aplicación.

- **Estructuras**: Se pueden ver las 3 estructuras navegables en el plano, cada una con su nombre que permitirá al usuario hacer la selección del destino. Se pueden ver los Puntos que determinarán su forma y dimensiones y el punto de navegación asociado a estas.

Podemos ver a continuación el plano resultante luego de definir estos datos:

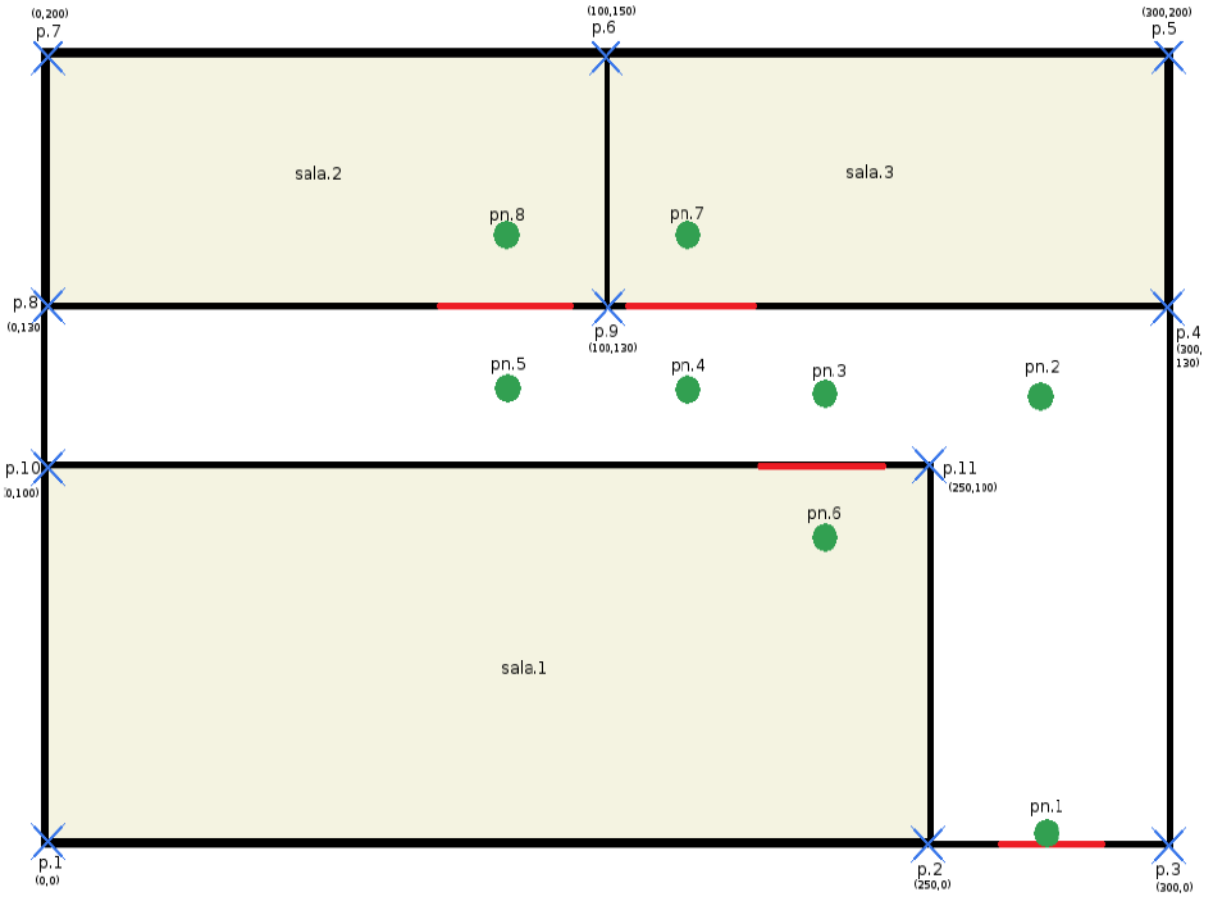


Figura 19. Edificio simplificado con información agregada

En la figura se puede ver la información más relevante que necesitaremos para instanciarlo. por ejemplo podemos ver como *sala.1* está formado por los puntos *p.1,p.2, p.10, p.11* , y que tiene asociado el punto de navegación *pn.6*

Diagrama de Instancias

En el siguiente diagrama de Instancias podemos ver cómo representamos el plano con nuestro modelo de clases, cada una de las instancias de las clases Estructura, PuntoDeNavegacion y Punto, y sus relaciones.

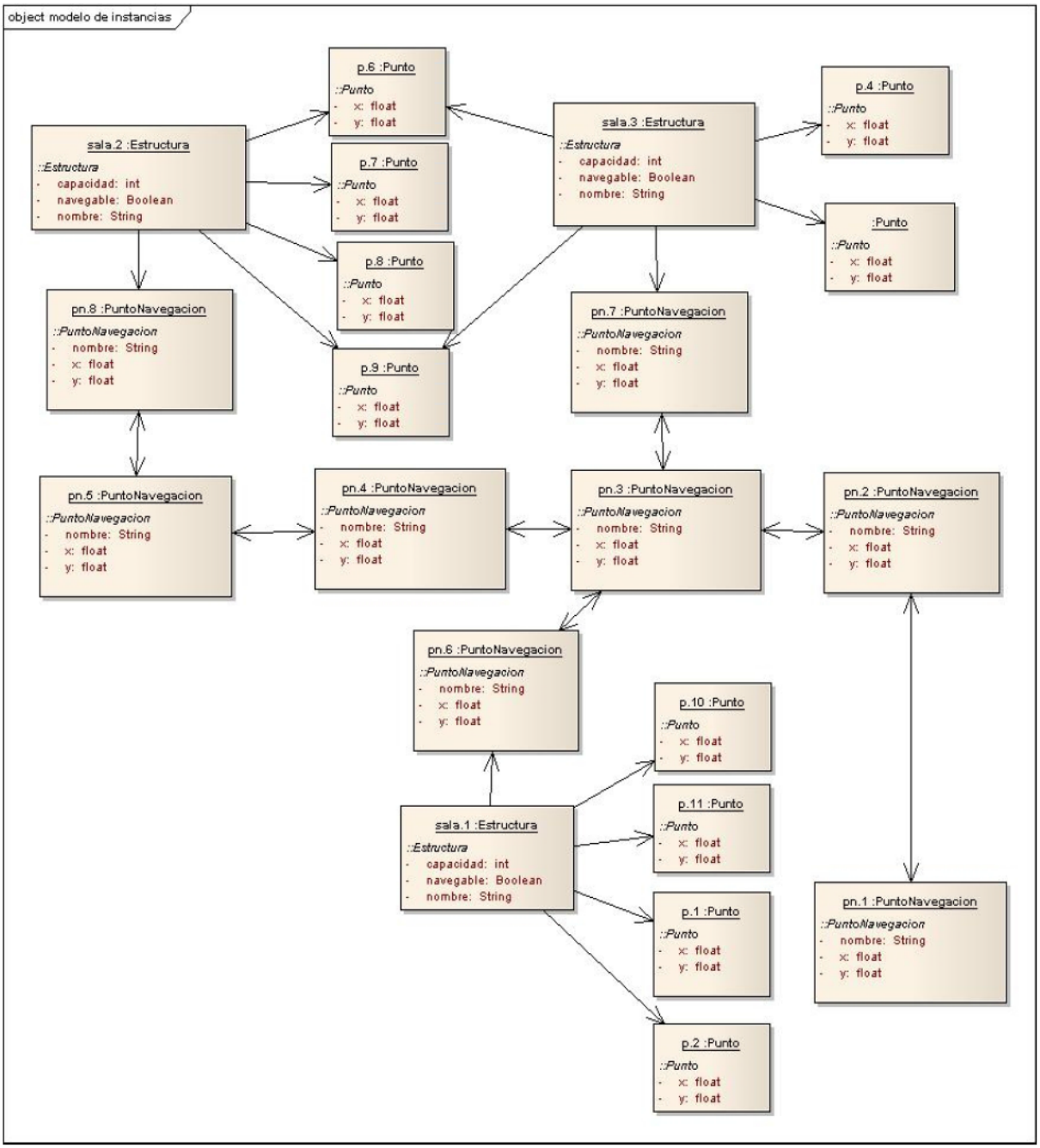


Figura 20. Diagrama de instanciación del edificio simplificado

El cálculo de camino utiliza el grafo que se forma de las relaciones entre los puntos de navegación, cada relación con otro punto de navegación contiene la distancia (hasta este), esta distancia es lo que utilizaremos para determinar el camino más corto entre dos puntos.

Grafo de Navegación

En la siguiente imagen se puede ver el grafo resultante de la instanciación de puntos de navegación definidos en el edificio de ejemplo.

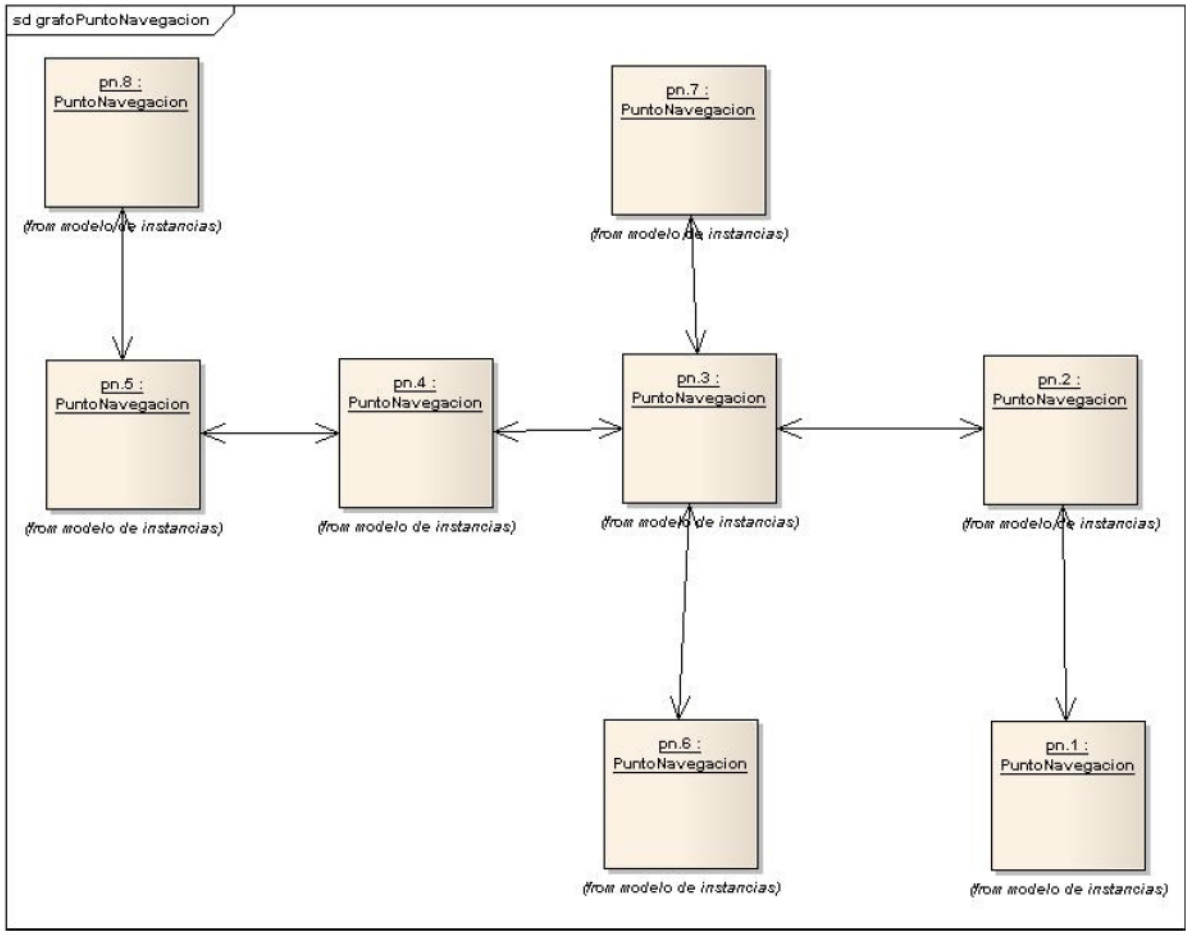


Figura 21. Grafo resultante de la instanciación del edificio simplificado

Ahora veremos el grafo marcado con azul en el plano del edificio, las distancias entre puntos de navegación determinan los costos de pasar de un nodo a otro.

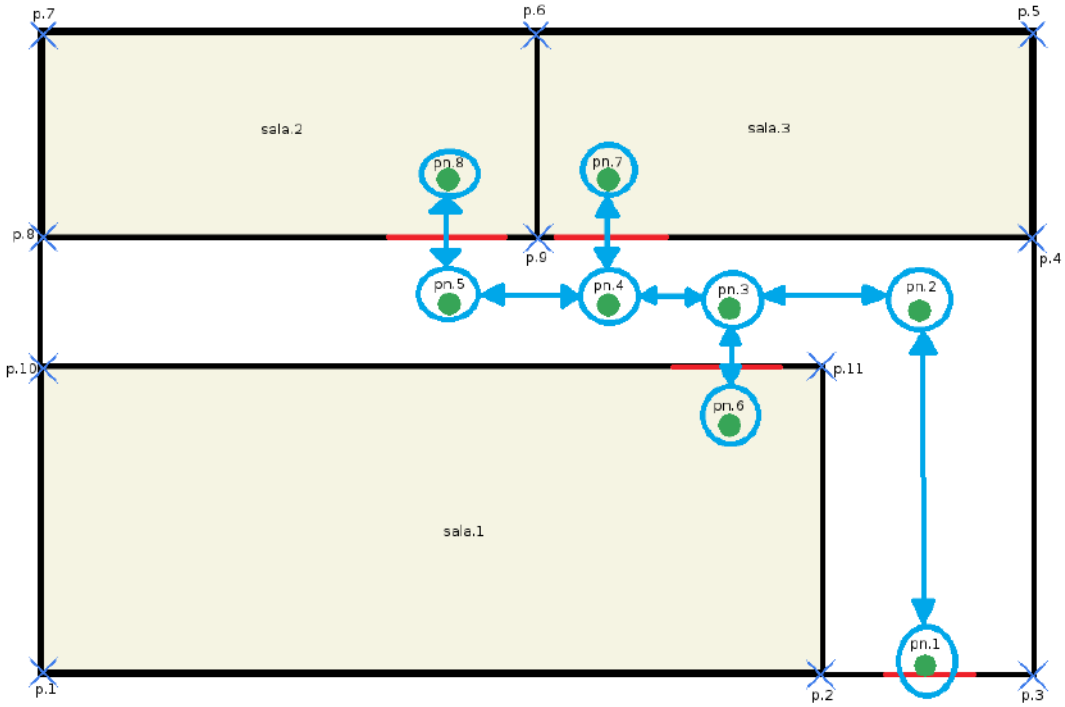


Figura 22. Grafo de puntos de navegación

6.4.Ejemplo navegación

Hemos analizado la instanciación del edificio navegable y como se define el grafo de caminos. Ahora analizaremos en este plano de ejemplo los casos de uso que utilizan estos datos para navegar.

Leer Posición Actual

Cada uno de los puntos de navegación (pn.1, pn.2, pn.3, pn.4, pn.5, pn.6, pn.7 y pn.8) estará codificado en un código QR que contendrá la url de la aplicación y el id del punto de navegación leído como parámetro, esto permitirá a la aplicación determinar la posición actual del usuario. Al leer el código QR el usuario decodifica la url y la utiliza para acceder al sistema mediante un browser en su dispositivo móvil.

Buscar Destino

Una vez determinada la posición actual del usuario, se le presentará una lista de estructuras navegables como posibles destinos, y podrá elegir hacia cual desea dirigirse. En este caso, la aplicación nos mostrará un listado que contedrá: *sala.1* , *sala.2* , *sala.3*.

Navegar a Destino

Ya contamos con un punto de navegacion de origen y el punto de navegacion asociado a la estructura que elegimos en el listado de “buscar destino”. La aplicación recupera todos los puntos de navegación y sus relaciones entre sí, de esta forma genera un grafo donde la distancia entre estos será utilizada para el cálculo del costo del camino. Se calcula el camino, y se muestra al usuario en pantalla la representación del edificio con el camino a seguir, su posición actual y su punto de destino.

A modo de ejemplo asumiremos que el usuario se encontraba en el punto de navegación pn.1 cuando leyó el código QR, y que en la búsqueda de destinos eligió sala.3.

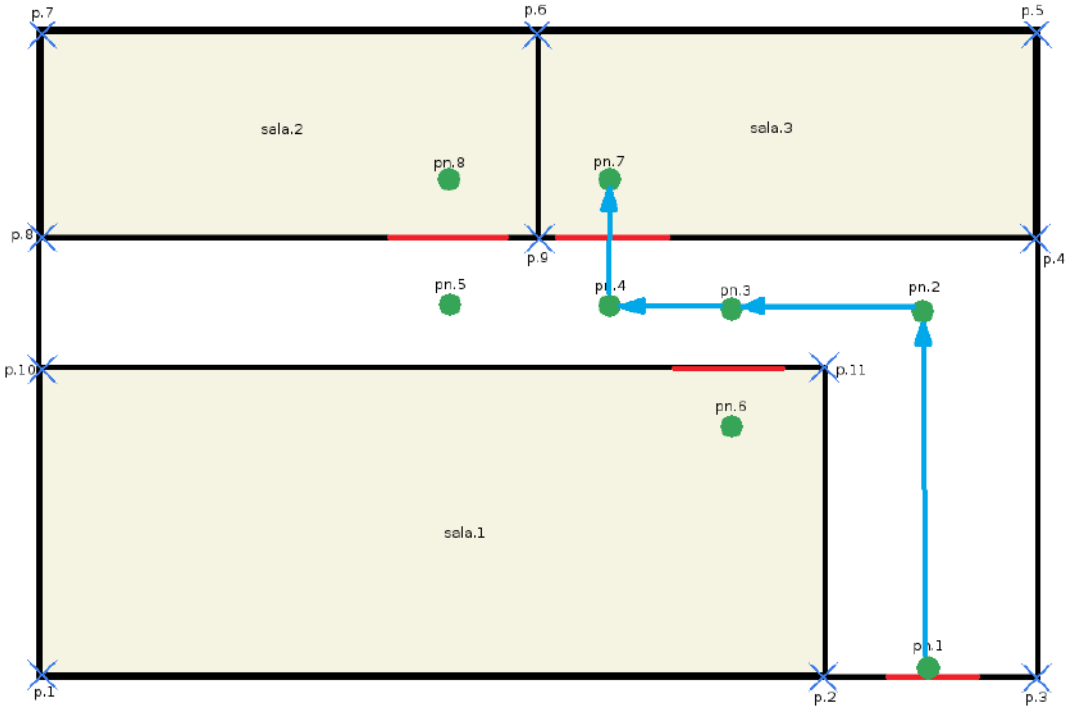


Figura 23. Ejemplo de cálculo de camino

Ver detalle de Estructura

Estando en la pantalla de navegación donde se representa el edificio y cada una de sus partes, el usuario puede seleccionar una estructura y ver detalles asociados a esta, como fotos, textos, videos, etc. La aplicación recibe la estructura seleccionada y muestra todos los datos asociados a ella.

Modificar Destino

Estando en la pantalla de navegación, el usuario puede modificar el destino en cualquier momento, seleccionando una estructura nueva en el mapa y marcando la misma como nuevo destino. El sistema responde de manera similar a “navegar a destino”, en este caso el punto de navegacion de origen será la ubicación actual del usuario, y el punto de navegación de destino será el asociado a la estructura seleccionada.

7.Aplicación Indoor-Navigation

El objetivo de este proyecto es construir una aplicación que permita guiar a los usuarios mediantes el uso de dispositivos móviles, desde el punto actual de este a un punto de destino dentro de un edificio.

Aplicaciones como Google Street View nos llevaron a pensar que sería interesante realizar una aplicación en la que los usuarios accediendo a través de sus dispositivos móviles, pudiesen “navegar” en un edificio, siendo guiados hacia su destino.

Esta situación nos plantea varios desafíos:

- Como determinar la ubicación del usuario en edificios cerrados.
- Como representar de manera intuitiva al usuario en el edificio, la ubicación actual obtenida, y el camino hacia su destino.
- Que la aplicación se pueda ejecutar en la mayor cantidad de dispositivos móviles.

La idea principal de nuestro trabajo es que los usuarios puedan ser guiados en un ambiente de interiores a través de dispositivos móviles, pero a su vez, que la aplicación pueda ser accedida por la mayor cantidad de móviles. Fue por esto que se decidió pensar en una solución cross-plataform, es decir, no desarrollar una aplicación para un sistema operativo móvil específico (Android, iOS, Symbian OS) ya que de este modo, dejaríamos dispositivos móviles fuera de la solución, fue por este motivo que se decidimos que la aplicación sea web, logrando de esta manera que cualquier móvil actual con un browser y conexión a internet puede acceder a la misma.

Otro aspecto importante de la aplicación es como será obtenida la ubicación actual del usuario dentro del edificio. Existen muchas formas de posicionar al usuario indoor con distintos grados de complejidad, nuestra postura sobre este punto fue la de abstraernos tanto como podamos de cómo se obtiene la ubicación del usuario y crear una aplicación en la cual la tecnología de posicionamiento pueda cambiar y la aplicación siga respondiendo de la misma manera, es decir, sea capaz de representar la posición del usuario en un mapa del edificio, mostrando el camino al destino que eligió y guiandolo en el camino.

Teniendo la aplicación desacoplada de la forma de captura de la posición del usuario, decidimos usar una de las más simples y fáciles de implementar. Codificamos ubicaciones de puntos de navegación en códigos QR, para que los usuarios puedan leerlos con sus dispositivos móviles y así poder informarle a la aplicación la ubicación actual del usuario. Existen aplicaciones que leen códigos QR disponibles para todas los Sistemas Operativos móviles existentes.

7.1.Edificio Facultad de Informática UNLP

Elegimos el Edificio de la Facultad de Informática sito en 50 y 120 como estructura a representar y navegar. Para esto conseguimos un plano de la primera etapa de construcción de la Facultad, que nos ayudó a definir cada una de las aulas. Las dimensiones no estaban especificadas en este plano por lo que definimos una escala, y con esta definimos coordenadas (x,y) para cada punto definido, y distancias entre puntos de navegación.

A continuación se puede ver el plano sobre el cual nos basamos para crear las estructuras y puntos de navegación:

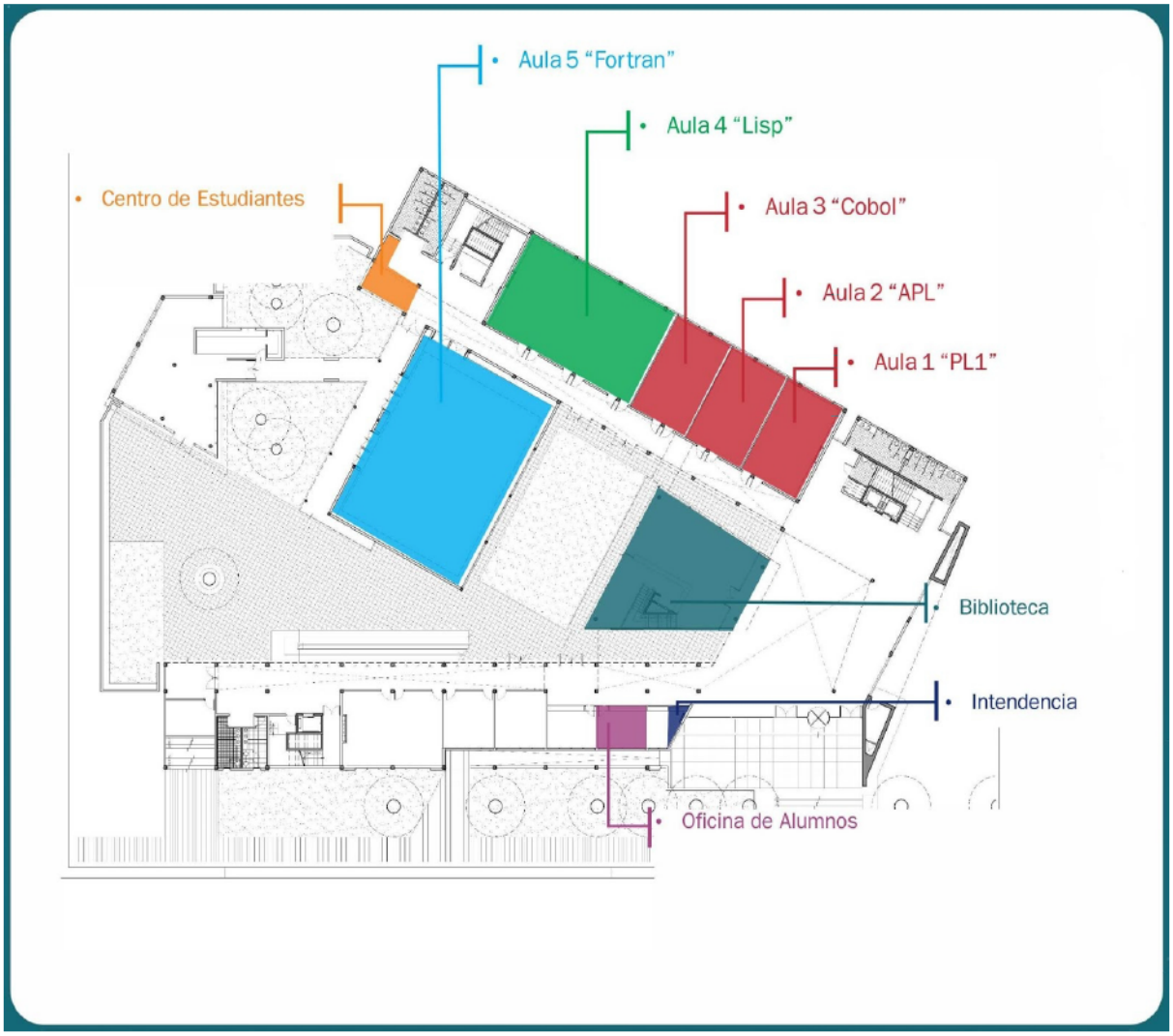


Figura 24. Plano del edificio de la Facultad de Informática

7.2.Instanciación de estructuras

Para la instanciación de los objetos (estructuras, puntos, puntos de navegación, etc) se diseñó e implementó una herramienta de edición de planos. Esta permite a un usuario registrado como administrador modificar el plano que luego utilizará la aplicación indoor-navigation.

Figura 25. Pantalla ingreso a la administración de planos

Creación de una estructura

A continuación mostraremos un ejemplo de como se realizo la carga de estructuras:

Figura 26. Creación de una nueva estructura

Creamos la estructura “Intendencia” la cual es del tipo “navegable”, que sea navegable nos da la pauta de que la misma tendrá un punto de navegación asociado.

Figura 27. Edición de una estructura

La herramienta también cuenta con la posibilidad de edición de objetos para el caso de que necesitemos modificar alguno de los atributos.

Creación de un punto

Figura 28. Creación de un nuevo Punto

Creamos un punto y lo relacionamos con la estructura **Intendencia**, como podemos observar el punto tiene como origen x = 2990 e y = 105. Estas coordenadas indican en qué posición del mapa queremos que el punto sea dibujado.

Figura 29. Listado de Puntos

Podemos observar el listado de puntos con los 3 puntos que creados para la estructura **Intendencia**.

Creación de Puntos de Navegación

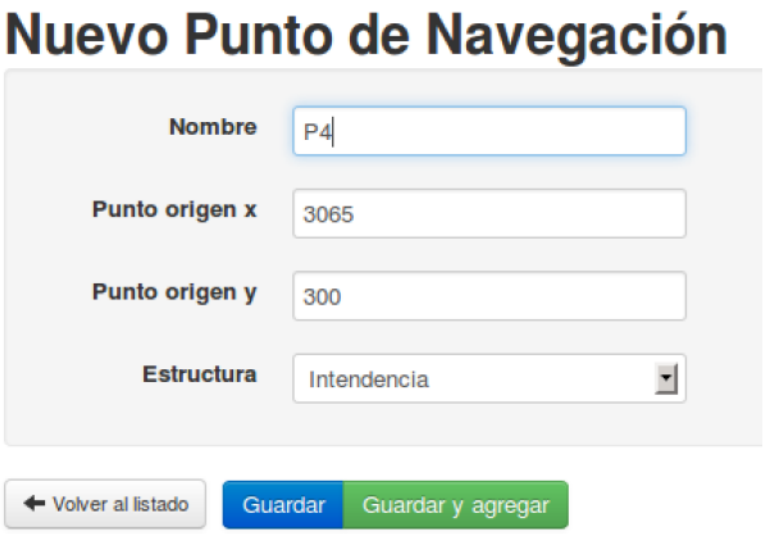


Figura 30. Creación de Punto de Navegación

Creamos el punto de navegación P4 con las coordenadas x = 3065 e y = 300, y lo relacionamos con la estructura **Intendencia**.

7.3.Navegación Indoor

Mostraremos a continuación la aplicación Indoor-Navigation desarrollada, y como esta resuelve cada uno de los casos de usos planteados.

Tomaremos como ejemplo la navegación de un usuario situado en la puerta de la facultad (punto de navegación con id=1) hasta el aula APL. Veremos la interacción del usuario con el sistema paso a paso y la funcionalidades que este le ofrece.

Leer Posición Actual

Teniendo nuestro modelo instanciado ya es posible navegar la facultad de informática. El ingreso a la aplicación se realiza a través de la lectura de Códigos QR, dentro de un código QR se encuentra codificada una URL con el parámetro **actual_id**, que le indica al Indoor Navigation la ubicación actual del usuario.

El siguiente ejemplo de código QR contiene codificada la URL:
http://localhost/indoor-navigation/index.php/buscar?actual_id=1

Figura 31. Código QR con url de la aplicación y punto de navegación codificada

La URL peticiona la acción **buscar** del Indoor Navigation con el parámetro (**actual_id** = **1**).
Existirá un código QR similar a éste para cada punto de navegación para situar al usuario, con el id de este asociado para determinar la ubicación actual.

Buscar Destino

Al abrirse la URL en el browser del dispositivo móvil del usuario se observa la siguiente pantalla de búsqueda:

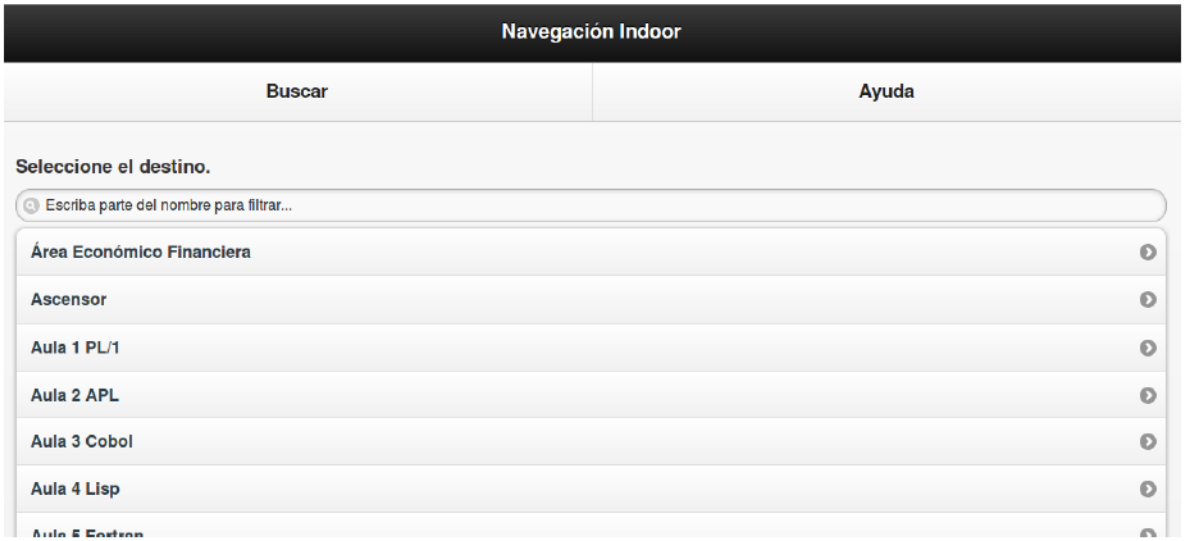


Figura 32. Pantalla de búsqueda de destino

La pantalla ofrece al usuario un listado con todas las estructuras navegables que posee la facultad, es decir, aquellas que tienen un punto de navegación asociado. El usuario puede filtrar los resultados realizando una búsqueda por nombre.

En el siguiente ejemplo, mostramos la busqueda del aula APL.

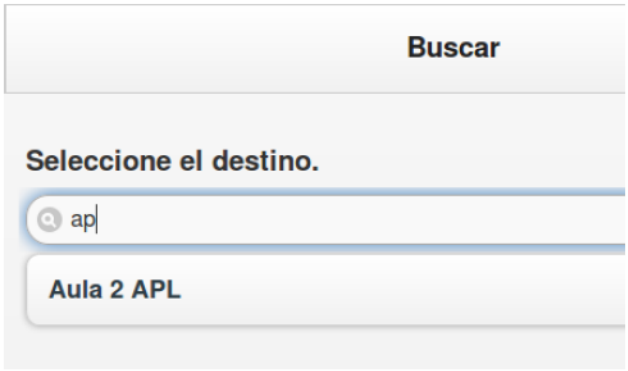


Figura 33. Ejemplo autocompletado de búsqueda

La interfaz fue creada utilizando jQueryMobile, que nos asegura que podrá ser visualizada correctamente en cualquier browser. La búsqueda de destinos tiene un autocomplete, con el cual el filtro de búsqueda se va refinando a medida que el usuario escribe su destino, ofreciendo resultados parciales que agilizan la selección de destino.

Navegar a Destino

Al realizar la petición sobre el Aula APL, el usuario es trasladado a la pantalla de navegación. Continuando con el ejemplo, el usuario está situado en el punto de navegación 1 (**actual_id** = **1**), este punto de navegación se corresponde con la puerta de la facultad de informática. La siguiente pantalla muestra la navegación desde la puerta de la facultad (lectura del código QR) hasta la selección realizada por el usuario (Aula 2 APL).

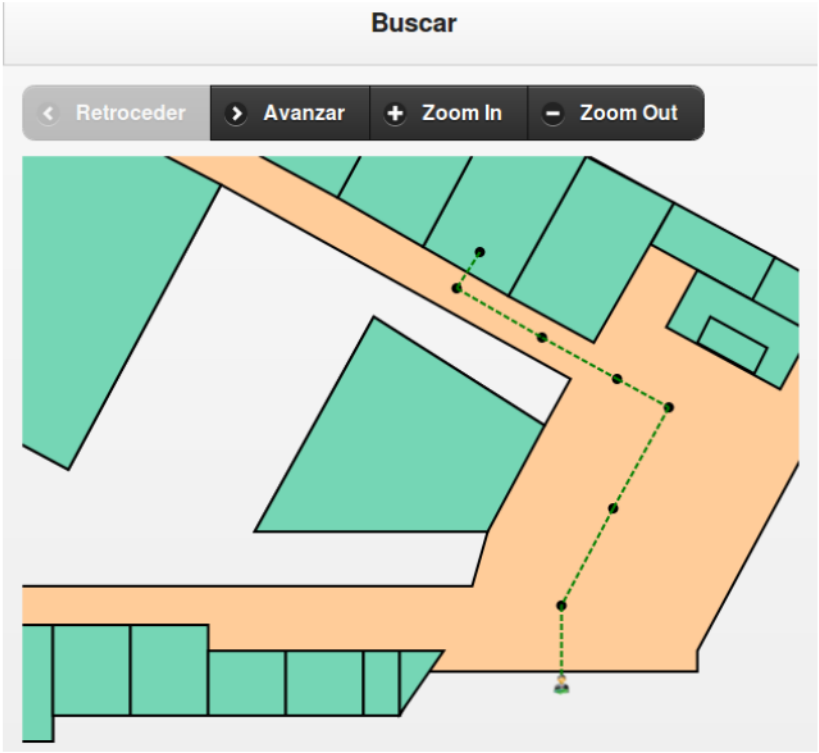


Figura 34. Ejemplo navegacion a destino seleccionado

Para poder dibujar el plano del edificio, camino, posición del usuario y puntos de navegación se utiliza el tag <canvas> de HTML5, y la librería javascript KineticJS.
La aplicación obtiene los datos de cada una de las estructuras instanciadas, puntos de navegación, calcula el camino y deja todos estos datos disponibles para que la capa de presentación los utilice.
Utilizando KineticJS creamos las estructuras, situando los vértices de esta con las coordenadas de los puntos asociados a la estructura a dibujar. El color de la estructura dependerá de si es navegable (color verde) o no (color salmón). A cada estructura creada se asigna un handler para manejar el evento “click” o “touch” , el cual luego pedirá al sistema más información sobre esta estructura.
El camino calculado por la aplicación está compuesto por una colección de puntos de navegación, para representarlas en el plano se recorre esta colección y se dibujan utilizando KinectJS en las coordenadas asociadas al punto de navegación y se dibuja con líneas punteadas la unión de los puntos de navegacion, formando asi el camino a seguir.
El primer punto de navegación de la colección representa la posición actual del usuario (el lugar donde leyó el código QR y desde donde inicia la navegación), su ubicación es representada por el icono de una persona.
El usuario a medida que avanza en el edificio real podrá seguir leyendo los códigos QR que encuentre en los puntos de navegación que atraviesa y así su posición en el mapa de la aplicación será actualizada.
Para simular la lectura de códigos QR contamos con una barra de navegación en la parte superior del mapa que nos permite “Avanzar” al siguiente punto de navegación en el camino y así actualizar su posición en la aplicación.

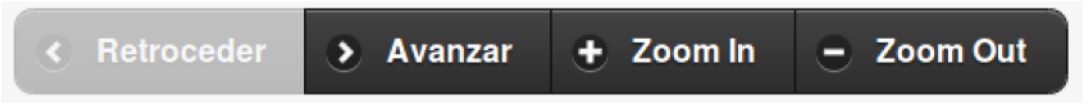


Figura 35. Barra de navegación

El botón retroceder, permite volver al punto de navegación anterior en el camino. en una primera instancia se encuentra deshabilitado ya que el usuario no puede retroceder debido a que se encuentra en el punto inicial de su navegación.
La siguiente pantalla muestra el Indoor Navigation luego de presionar el botón avanzar una vez.

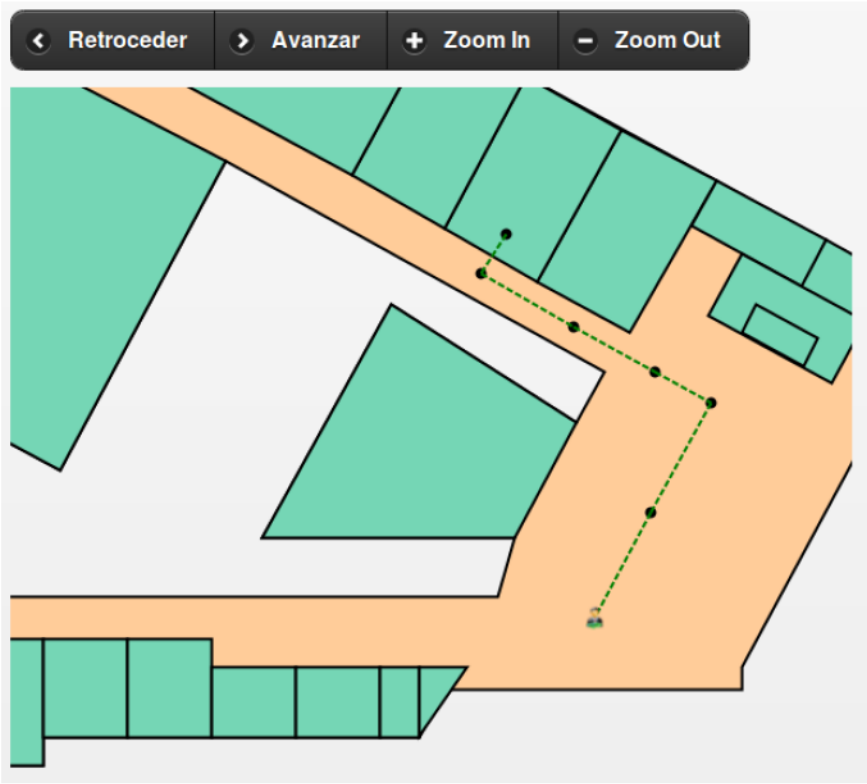


Figura 36. Simulación de navegación al presionar la opción "Avanzar"

Cada vez que el usuario modifica su posición, el plano es redibujado.
La barra de navegación nos permite hacer zoomIn y zoomOut, existen 5 niveles de zoom en la aplicación, por defecto se utiliza el intermedio, hay 2 niveles de zoomOut y 2 niveles de zoomIn. Si el usuario desea hacer un zoomIn por ejemplo, el mapa del edificio es redibujado en su totalidad utilizando otra escala en las dimensiones de las estructuras y en los valores de las coordenadas. Esto permite al usuario visualizar la totalidad del edificio haciendo zoomOut o centrarse en su posición haciendo zoomIn.
Ejemplo de la aplicación con el nivel de zoomIn máximo:

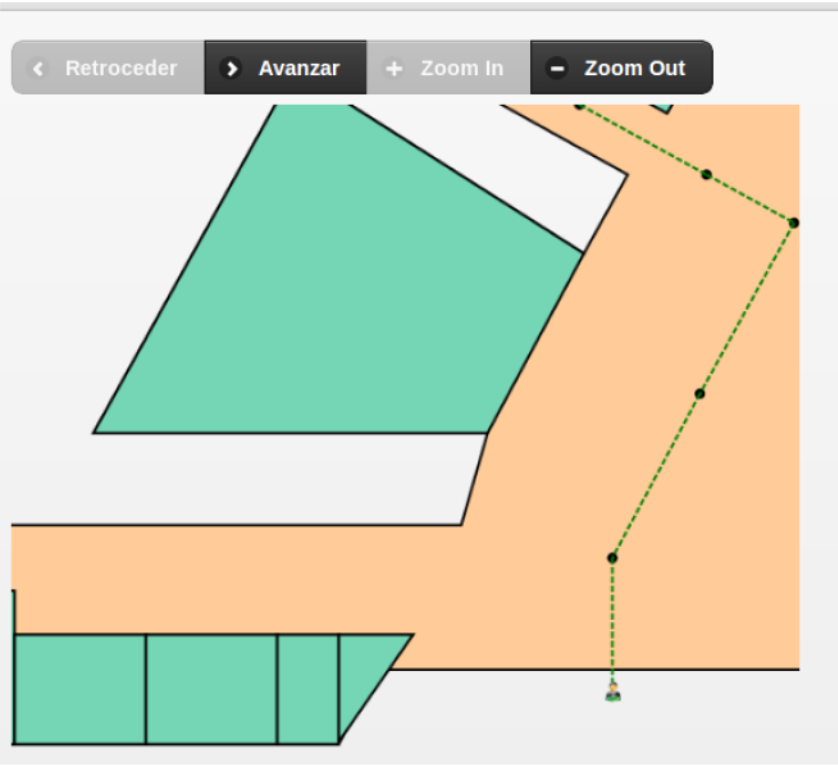


Figura 37. ZoomIn en la aplicación

Ejemplo de la aplicación con el zoomOut al máximo:

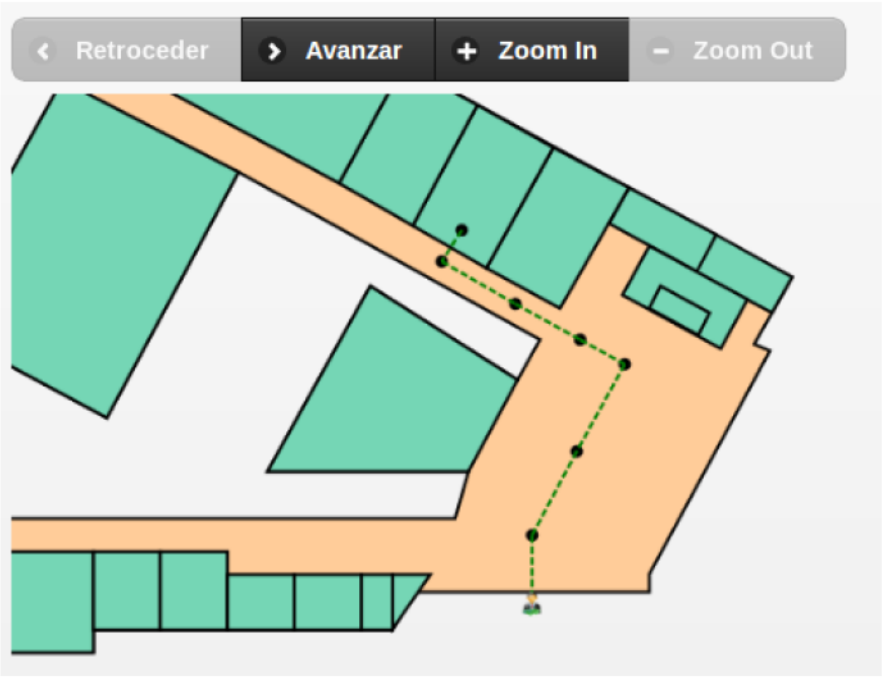


Figura 38. ZoomOut en la aplicación

Ver detalle de Estructura

Durante la navegación, es posible consultar las distintas estructuras. Al realizar un click sobre una estructura navegable, obtendremos un detalle de la misma.
La siguiente pantalla de ejemplo, nos muestra el detalle de realizar click sobre la Biblioteca:

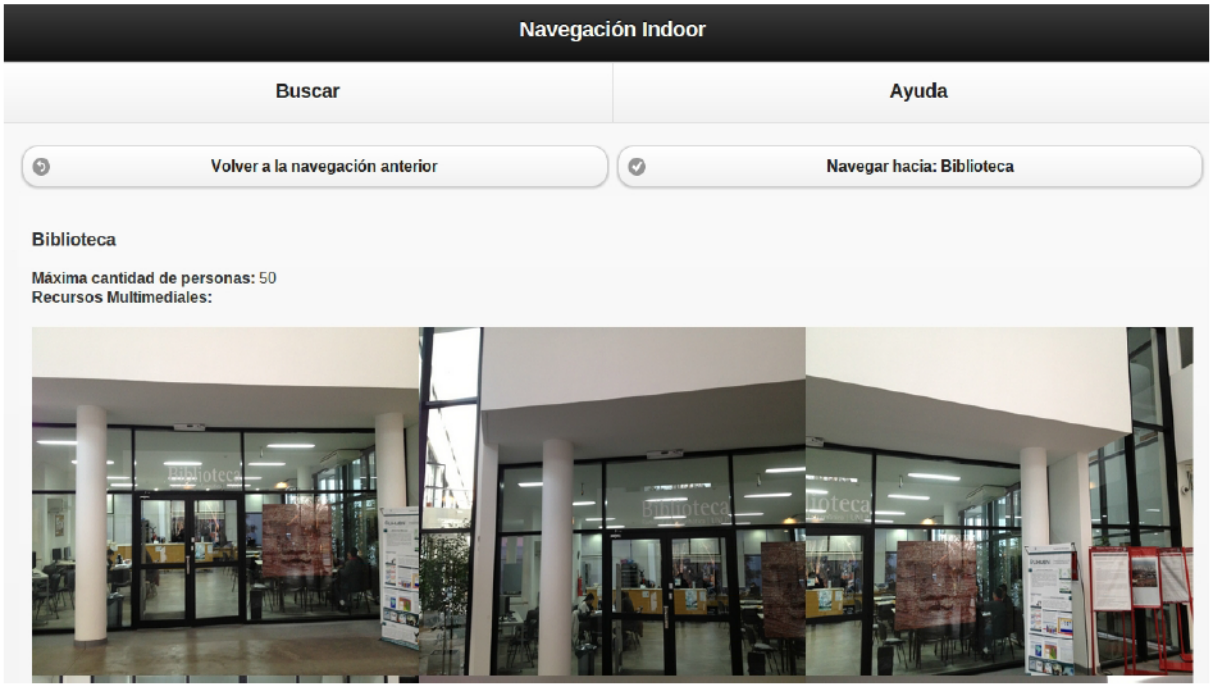


Figura 39. Detalle de una estructura

En el detalle de la estructura se pueden ver fotos, textos y demás información asociada.
En este caso al tratarse de la facultad y sus aulas un ejemplo de información a mostrar en el detalle, podría ser los horarios de las cursadas en esta aula. En el caso de la biblioteca se muestran imágenes correspondientes a la misma. Cada una de estas imágenes es ampliable, mostrando la imagen seleccionada en una resolución mayor en un popup.

Modificar Destino

En está pantalla también se da la opción de seleccionar esta estructura como nuevo destino, haciendo que el sistema realice el cálculo del camino nuevamente entre la estructura actual seleccionada y la posición actual del usuario.
Podemos ver en la siguiente imagen como si hacemos click en “Navegar hacia: Biblioteca” la aplicación nos muestra nuevamente el mapa, con un nuevo camino calculado, esta vez hacia el nuevo destino.

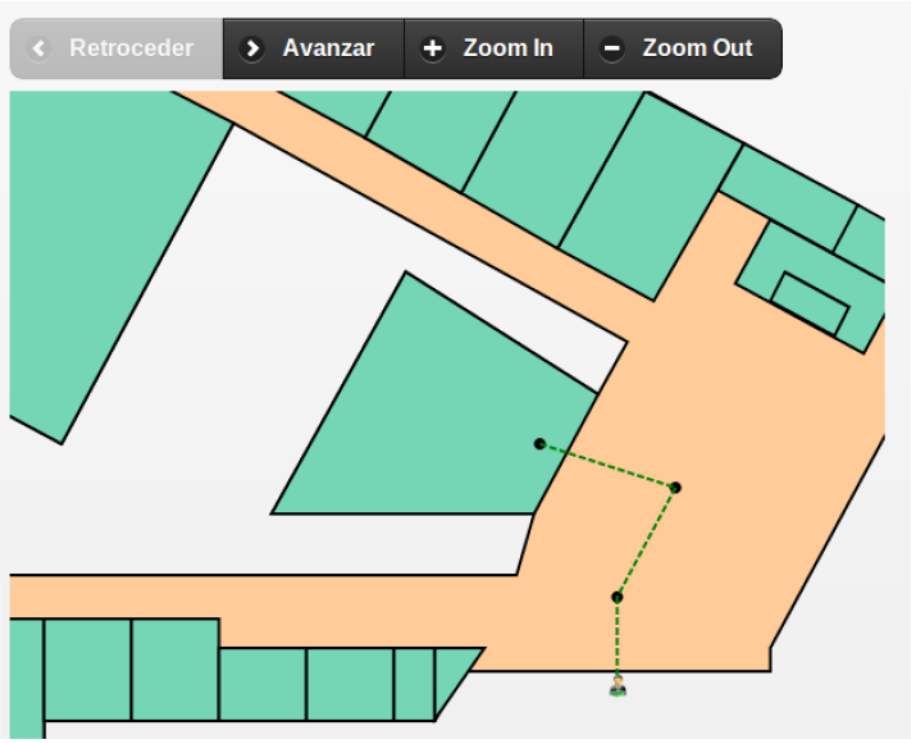


Figura 40. Ejemplo de modificación de destino actual

Si avanzamos siguiendo el camino trazado por la aplicación, al llegar al punto de navegación más cercano a la estructura de destino nos mostrará un mensaje indicándonos que hemos llegado a destino y 2 opciones a seleccionar:

- Cerrar el popup y ver el mapa para elegir un nuevo destino en el mapa, abriendo el detalle de una estructura y marcando la misma como nuevo destino.
- Ir a la seccion de Busqueda de destino y comenzar una nueva navegación, esta vez la posición actual del usuario será la estructura a la que se llegó.

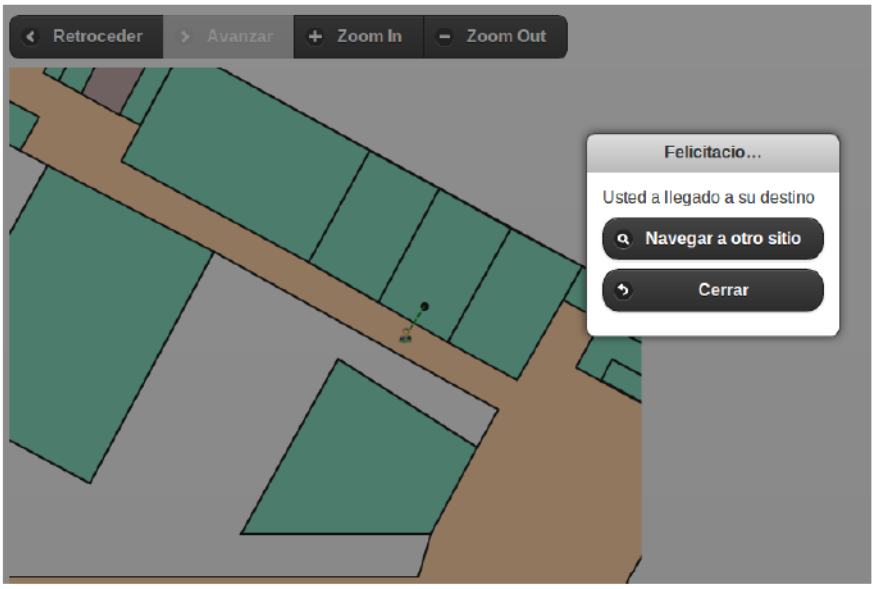


Figura 41. Mensaje informativo de llegada a destino

8.Navegación 3D

Al comienzo del desarrollo del prototipo, la interfaz 3D fue la que mayor esfuerzo requirió y donde hicimos mayor foco. Comenzamos buscando y leyendo que estaba hecho y que se podíamos hacer, y a medida que fuimos progresando en el desarrollo nos fuimos encontrando con cada vez mayores obstáculos, que a la larga nos hicieron optar por desarrollar más funcionalidad en la interfaz 2D, dejando al 3D como posible trabajo futuro.

El desarrollo de esta interfaz está basado en el primer modelo, y en el edificio de la Facultad de Informática de 120 y 50. Es por esto que las dimensiones y formas de las aulas están determinadas por el alto, ancho. Cuando surgió el cambio de edificio, adaptar lo ya desarrollado en 3D al nuevo modelo fue imposible.

A continuación describimos nuestra experiencia con el contexto 3D del elemento canvas de HTML5.

Prototipo 3D

Para la creación de la interfaz 3D, comenzamos detallando cuál fue el objetivo planteado. La idea fue crear una interfaz 3D en la que podamos situar al usuario dentro del edificio, en este caso la antigua Facultad de Informática. Es importante recordar que el desarrollo 3D esta basado en el modelo de entidad anterior al actual, en el cual cada estructura estaba definida por un ancho y un alto, y el edificio de la facultad representado es el situado en calle 50 y 115.

Para esto necesitamos representar el primer piso de la facultad en tres dimensiones, donde crearemos un ambiente grande formado por 4 paredes, un piso y un techo, y dentro de este crear polígonos que representen las aulas o estructuras. A cada aula o estructura se le puede aplicar una textura asociada a ésta, siendo esta textura una fotografía de su fachada exterior, con lo cual servirá al usuario a identificarlas. Por último se representará el punto de vista del usuario, teniendo este un ángulo de visión, a una distancia del piso y con la posibilidad de moverse por el edificio voluntariamente utilizando controles de desplazamiento, o bien situandolo en un punto del edificio determinado al leer un código QR.

Explicación de la implementación

Definición del elemento canvas.

```
<body>
    <canvas id="myCanvas" width="1366" height="610">
    </canvas>
</body>
```

Pasamos a definir las variables globales al script que necesitaremos. Las mas importantes son:

- **alturaY:** representa la altura del edificio, como las estructuras tienen información de sus dimensiones en 2D (alto y ancho), la distancia del suelo al techo esta dada por este valor y es usado en todas las estructuras creadas.
- **shapeBuffer** y **shapePosition**: estos dos arreglos son utilizados para guardar la información de como crear la estructura 3D (shapeBuffer) y su posición dentro del espacio 3D.
- **cubeBuffers** , **floorBuffer**, **wallBuffer:** estas tres variables contienen la información para crear el ambiente más grande que servirá para contener a las estructuras creadas.
- **textures:** contiene la textura utilizada para cada una de las estructuras creadas y almacenadas en el shapeBuffer.
- **camera:** esta variable contiene la posición del punto de vista actual, con su altura , ángulo, etc.
- **speed:** permite determinar la velocidad de desplazamiento del usuario dentro del ambiente 3D.

Quedan muchas otras variables utilizadas para manejar los eventos de las teclas, movimiento, ángulo de la visión, etc

```
<script type="text/javascript">

    //tratar de usar estas variables donde se pueda!
    var totalBuildingWidht = 50;
    var totalBuildingHeight = 50;

    var kin;
    var cubeBuffers = {};
    var floorBuffers = {};
    var wallBuffers = {};
    var angle = 0;
    var textures = {};
    var shapePositions = [];

    //shape variables and constants
    var shapeBuffers = [];
    var ALTURAY = 5; //es la altura del piso al techo, cambiar esto y se modifica la de todos los shape
    var shapePositions = [];

    // movements
    var STILL = "STILL";
    var FORWARD = "FORWARD";
    var BACKWARD = "BACKWARD";
    var LEFT = "LEFT";
    var RIGHT = "RIGHT";

    var straightMovement = STILL;
    var sideMovement = STILL;
    var speed = 8; // units per second
    // camera
    var mouseDownPos = null;
    var mouseDownPitch = 0;
    var mouseDownYaw = 0;
    var camera = {
        x: 0,
        y: 1.5,
        z: 5,
        pitch: 0,
        yaw: 0
    };
```

Ya habiendo definido las variables necesarias para contener la información de las estructuras que crearemos, definimos la función que inicializa el comportamiento del script.

window.onload() hace que el código definido en esta función se ejecute cuando la página termine de cargar el DOM.

Obtenemos el elemento canvas definido anteriormente, e inicializamos el contexto. Para esto utilizamos una librería llamada **kinetic3D**, la cual nos abstrae en ciertos puntos de operaciones matemáticas, y tiene la ventaja de poder agrupar toda la información de una estructura en un objeto, con lo cual hace que su manipulación luego del momento de creación sea más fácil.

Cuando se hace ***new Kinetic_3d("myCanvas")*** esta librería lo que hace es crear un objeto, en el cual obtiene el contexto webGL del canvas. A continuación se pueden ver las primeras 3 líneas del código de inicialización del objeto kinetic:

```
var Kinetic_3d = function(canvasId){
  this.canvas = document.getElementById(canvasId);
  this.context = this.canvas.getContext("experimental-webgl");
  this.drawStage = undefined;
```

Utilizamos la versión 1.0.0 de Kinetic , que en el momento de desarrollo de la interface 3D era la última. Actualmente esta en la versión 4.2, lo que indica la rápida evolución de la librería.

Una vez que obtenemos el contexto webGL, se inicializan los manejadores de eventos (clicks, teclas, etc) y los buffers son inicializados. También se cargan los paths correspondiente a cada una de las texturas a utilizar.

```
window.onload = function(){
    var canvas = document.getElementById("myCanvas");
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;

    kin = new Kinetic_3d("myCanvas");
    kin.setShaderProgram("TEXTURE_DIRECTIONAL_LIGHTING");

    attachListeners();
    //alert("1");
    initBuffers();

    //inicializar posiciones de los Shape

    var sources = {
        shape: "indoor3d_files/wall2.jpg",
        metalFloor: "indoor3d_files/metalFloor.jpg",
        metalWall: "indoor3d_files/wall1.jpg",
        ceiling: "indoor3d_files/ceiling.jpg"
    };

    loadTextures(sources, startAnimation);
};
```

Esta función inicializa de los manejadores de eventos. Se puede ver que trata los eventos del mouse y teclas, en un dispositivo móvil la librería Kinetic convierte el gesto touch a estos eventos, por lo que podemos abstraernos de los mismos.

```
function attachListeners(){
    kin.getCanvas().onmousedown = function(){
        handleMouseDown();
    };
    kin.getCanvas().onmousemove = function(){
        handleMouseMove();
    };
    document.body.onmouseup = function(){
        handleMouseUp();
    };
    document.body.onmouseout = function(){
        handleMouseUp(); // same as mouse up functionality
    };
    document.body.onkeydown = function(evt){
        handleKeyDown(evt);
    };
    document.body.onkeyup = function(evt){
        handleKeyUp(evt);
    };

}
```

La función *initBuffer* es una de las más importantes, en ella estarán las definiciones y datos de cada una de las estructuras a crear. Se puede ver que en el código del prototipo se llama a la creación de 4 estructuras , esta es la única sección de todo el código que veremos que hay que modificar para que cargue todas las estructuras de la facultad, recibiría por parámetro un arreglo con las dimensión y posición de cada una de las estructuras,iteraría sobre este y llamaría a la función *createShape()* .

Una vez creada las estructuras (que en el prototipo crearemos 4) se llaman a las funciones que cargan los buffers del piso y las paredes de la estructura contenedora.

```
function initBuffers(){

    //Crea todas las shapes
    //alert("llamo a createShapes");
    createShape(5, 5, 0, 0);
    createShape(7, 5, 0, 9.1);
    createShape(3, 5, 0, 21);

    createShape(5, 5, 5.1, 0);
    //createShape(2.5,7.5, 0, 0);
    //createShape(7.6,5, 7.6, 2.5);

    initFloorBuffers();
    initWallBuffers();
}
```

La función *createShape* es la que crea y posiciona las estructuras, fue creada para adaptarse a los datos que se tenían sobre cada una de las estructuras en 2D, es decir que recibe el alto y el ancho, y un punto definido por un vertice de la misma. Más adelante veremos cómo convertimos las coordenadas 2D a un espacio 3D.

La posición y la forma de la estructura son guardadas en arreglos o buffers separados. Es en esta función donde se puede ver que el alto (distancia del piso al techo) nunca es indicada, esta distancia está determinada por la variable global que ya describimos.

```
function createShape(alto, ancho, pos_x, pos_y)
{
    //alert("adentro de createShapes");
    //en este buffer almaceno los datos de la figura a dibujar actual, y una vez que tengo todo armado
    // la almaceno en el buffer de Shapes gral.
    var thisShapeLocalBuffer = {};

    initLocalShapeBuffers(alto,ancho, thisShapeLocalBuffer);
    shapeBuffers.push(thisShapeLocalBuffer);
    initShapePositions(alto, ancho, pos_x, pos_y)
}
```

La funcion *initLocalShapeBuffer* es llamada desde *createShape*, recibe 2 parámetros, como ancho y alto, y a partir de estos define las matrices con los valores que determinarán la forma de la estructura y los almacena en el buffer.

```
function initLocalShapeBuffers(alto,ancho, thisShapeLocalBuffer) {

    var witdh = ancho ; //mitad del width original por el eje 0 que divide
    var height = alto; //lo mismo que el width, /2

    thisShapeLocalBuffer.positionBuffer = kin.createArrayBuffer([
        // Front face    x, y, z   -> y queda fija, hasta el techo, x ancho z= largo?
        -witdh, 0, height,
        witdh, 0, height,
        witdh, ALTURAY, height,
        -witdh, ALTURAY, height,

        // Back face
        -witdh, 0, -height,
        -witdh, ALTURAY, -height,
        witdh,  ALTURAY, -height,
        witdh, 0, -height,

        // Top face
        -witdh, ALTURAY, -height,
        -witdh, ALTURAY, height,
        witdh, ALTURAY, height,
        witdh, ALTURAY, -height,

        // Bottom face
        -witdh, 0, -height,
        witdh, 0, -height,
        witdh, 0, height,
        -witdh, 0, height,

        // Right face
        witdh, 0, -height,
        witdh, ALTURAY, -height,
        witdh, ALTURAY, height,
        witdh, 0, height,

        // Left face
        -witdh, 0, -height,
        -witdh, 0, height,
        -witdh, ALTURAY, height,
        -witdh, ALTURAY, -height
    ]);
```

```
thisShapeLocalBuffer.textureBuffer = kin.createArrayBuffer([
    // Front face
    0, 0,
    1, 0,
    1, 1,
    0, 1,

    // Back face
    1, 0,
    1, 1,
    0, 1,
    0, 0,

    // Top face
    0, 1,
    0, 0,
    1, 0,
    1, 1,

    // Bottom face
    1, 1,
    0, 1,
    0, 0,
    1, 0,

    // Right face
    1, 0,
    1, 1,
    0, 1,
    0, 0,

    // Left face
    0, 0,
    1, 0,
    1, 1,
    0, 1
]);
```

```
        thisShapeLocalBuffer.indexBuffer = kin.createElementArrayBuffer([
            0, 1, 2,        0, 2, 3, // Front face
            4, 5, 6,        4, 6, 7, // Back face
            8, 9, 10,       8, 10, 11, // Top face
            12, 13, 14,     12, 14, 15, // Bottom face
            16, 17, 18,     16, 18, 19, // Right face
            20, 21, 22,     20, 22, 23 // Left face
        ]);
    }
```

La función *initShapePosition()* determina la posición de la estructura creada en el espacio tridimensional. La funcion *initLocalShapeBuffer* es llamada desde *createShape*, recibe 2 parámetros, como ancho y alto, y a partir de estos define las matrices con los valores que determinarán la forma de la estructura y los almacena en el buffer.

Se pueden ver las transformaciones que sufren las coordenadas 2D para poder ser representadas en el espacio 3D, también como se tiene que determinar el ángulo de dibujo, este ángulo está expresado en radianes.

```
function initShapePositions(alto, ancho, pos_x, pos_y){
    shapeRange = 45;

    /*
        |valor Z
        |
--------|--------- valor X   |mirando para abajo aparece
        |
        |
    Valor y es a la altura del piso que esta el objeto.  */
    var x3d= -50 + (pos_x *2+ ancho);//xValueIn3dContext(pos_x) ;//+ (alto);
    var z3d= 50 - (pos_y *2+ alto );//zValueIn3dContext(pos_y) +(alto /2);
    alert("ancho = "+ancho+" alto=  "+alto+" x3d quedo= "+x3d+" y3d quedo="+z3d);
    var thisShapePosition = {};
    thisShapePosition .x = x3d;
    thisShapePosition .y = 0;
    thisShapePosition .z =  z3d;
    thisShapePosition .rotationY =  0;//Math.random() * Math.PI * 2;
    shapePositions.push(thisShapePosition );
}
```

Las 2 funciones que se pueden ver a continuación hacen la conversión de la que hablamos anteriormente, reciben posiciones correspondientes a un plano 2D y retornan posiciones en un espacio 3D.

```
function xValueIn3dContext(pos_x)
{
    /* convierte una pos 2d en una valida para el contexto 3d */
    var half = totalBuildingWidht -4;
    return -half + pos_x;
}

function zValueIn3dContext(pos_y)
{
    /* convierte una pos 2d en una valida para el contexto 3d */
    var half = totalBuildingHeight -4;
    return half - pos_y;
}
```

La función *initFloorBuffer()* inicializa el buffer con los datos correspondientes para poder dibujar el piso de la estructura contenedora.

```
function initFloorBuffers() {
    floorBuffers.positionBuffer = kin.createArrayBuffer([
        -50, 0, -50,
        -50, 0, 50,
        50, 0, 50,
        50, 0, -50
    ]);

    floorBuffers.textureBuffer = kin.createArrayBuffer([
        0, 25,
        0, 0,
        25, 0,
        25, 25
    ]);

    floorBuffers.indexBuffer = kin.createElementArrayBuffer([
        0, 1, 2,        0, 2, 3
    ]);

    // floor normal points upwards
    floorBuffers.normalBuffer = kin.createArrayBuffer([
        0,  1,  0,
        0,  1,  0,
        0,  1,  0,
        0,  1,  0
    ]);
}
```

initWallBuffers() hace lo mismo, pero inicializa las paredes de la estructura contenedora.

```
function initWallBuffers() {
    wallBuffers.positionBuffer = kin.createArrayBuffer([
        -50, ALTURAY, 0,
        50, ALTURAY, 0,
        50, -ALTURAY, 0,
        -50, -ALTURAY, 0
    ]);

    wallBuffers.textureBuffer = kin.createArrayBuffer([
        0, 0,
        25, 0,
        25, 1.5,
        0, 1.5
    ]);

    wallBuffers.indexBuffer = kin.createElementArrayBuffer([
        0, 1, 2,        0, 2, 3
    ]);

    // floor normal points upwards
    wallBuffers.normalBuffer = kin.createArrayBuffer([
        0,  0,  1,
        0,  0,  1,
        0,  0,  1,
        0,  0,  1
    ]);
}
```

Las siguientes 3 funciones como su nombre lo indica dibujan el piso , el techo y paredes de la estructura contenedora con los datos que contienen los buffers de cada una. También le asignan la textura definida para cada una

```
    function drawFloor(){
        kin.save();
        //kin.translate(0, -1.1, 0);
        kin.translate(0, 0, 0);
        kin.pushPositionBuffer(floorBuffers);
        kin.pushNormalBuffer(floorBuffers);
        kin.pushTextureBuffer(floorBuffers, textures.metalFloor);
        kin.pushIndexBuffer(floorBuffers);
        kin.drawElements(floorBuffers);
        kin.restore();
    }

    function drawCeiling(){
        kin.save();
        kin.translate(0, ALTURAY, 0);
        // use floor buffers with ceiling texture
        kin.pushPositionBuffer(floorBuffers);
        kin.pushNormalBuffer(floorBuffers);
        kin.pushTextureBuffer(floorBuffers, textures.ceiling);
        kin.pushIndexBuffer(floorBuffers);
        kin.drawElements(floorBuffers);
        kin.restore();
    }
```

```
function drawWalls(){
    kin.save();
    kin.translate(0, 3.9, -50);
    kin.pushPositionBuffer(wallBuffers);
    kin.pushNormalBuffer(wallBuffers);
    kin.pushTextureBuffer(wallBuffers, textures.metalWall);
    kin.pushIndexBuffer(wallBuffers);
    kin.drawElements(wallBuffers);
    kin.restore();

    kin.save();
    kin.translate(0, 3.9, 50);
    kin.rotate(Math.PI, 0, 1, 0);
    kin.pushPositionBuffer(wallBuffers);
    kin.pushNormalBuffer(wallBuffers);
    kin.pushTextureBuffer(wallBuffers, textures.metalWall);
    kin.pushIndexBuffer(wallBuffers);
    kin.drawElements(wallBuffers);
    kin.restore();

    kin.save();
    kin.translate(50, 3.9, 0);
    kin.rotate(Math.PI * 1.5, 0, 1, 0);
    kin.pushPositionBuffer(wallBuffers);
    kin.pushNormalBuffer(wallBuffers);
    kin.pushTextureBuffer(wallBuffers, textures.metalWall);
    kin.pushIndexBuffer(wallBuffers);
    kin.drawElements(wallBuffers);
    kin.restore();

    kin.save();
    kin.translate(-50, 3.9, 0);
    kin.rotate(Math.PI / 2, 0, 1, 0);
    kin.pushPositionBuffer(wallBuffers);
    kin.pushNormalBuffer(wallBuffers);
    kin.pushTextureBuffer(wallBuffers, textures.metalWall);
    kin.pushIndexBuffer(wallBuffers);
    kin.drawElements(wallBuffers);
    kin.restore();
}
```

drawStage() es la función que contiene los llamados a las funciones explicadas anteriormente. Lo hace actualizando la posición de la cámara (o punto de vista del usuario) y guardando el contexto.

```
function drawStage(){
    updateCameraPos();
    kin.clear();
    // set field of view at 45 degrees
    // set viewing range between 0.1 and 100 units away.
    kin.perspective(45, 0.1, 150.0);
    kin.identity();

    kin.rotate(-camera.pitch, 1, 0, 0);
    kin.rotate(-camera.yaw, 0, 1, 0);
    kin.translate(-camera.x, -camera.y, -camera.z);

    // enable lighting
    kin.enableLighting();
    kin.setAmbientLighting(0.5, 0.5, 0.5);
    kin.setDirectionalLighting(-0.25, -0.25, -1, 0.8, 0.8, 0.8);

    drawFloor();
    drawWalls();
    drawCeiling();
    drawShape();
}
```

drawShape() itera sobre el arreglo de posiciones ya cargado, y por cada posición saca del *shapeBuffer* una estructura con sus datos, le asigna una textura, y un ángulo de dibujo, la dibuja y guarda el contexto.

```
function drawShape(){
    //alert("drawShape");
    //alert("contenido de la var shapeBuffers[]0"+shapeBuffers[0]);
    for (var n = 0; n < shapePositions.length; n++) {
        kin.save();
        //var thisShapePosition = shapePositions[n];
        var thisShapePosition = shapePositions[n];
        var thisShapeBuffers = shapeBuffers[n];
        kin.translate(thisShapePosition.x, thisShapePosition.y, thisShapePosition.z);
        kin.rotate(thisShapePosition.rotationY, 0, 1, 0);
        kin.pushPositionBuffer(thisShapeBuffers);
        kin.pushNormalBuffer(thisShapeBuffers);
        kin.pushTextureBuffer(thisShapeBuffers, textures.shape);
        kin.pushIndexBuffer(thisShapeBuffers);
        kin.drawElements(thisShapeBuffers);
        kin.restore();
    }
}
```

La función *loadTextures()* carga las texturas y las renderiza. Las texturas son imágenes con formato png que se repiten al ser asignadas a la “pared” de una estructura.

Si como textura para una estructura utilizamos una foto de su fachada, cada una de las estructuras sería más representativa, y daría al usuario la sensación de encontrarse dentro de la facultad.

```
function loadTextures(sources, callback){
    var loadedImages = 0;
    var numImages = 0;
    for (var src in sources) {
        // anonymous function to induce scope
        (function(){
            var key = src;
            numImages++;
            textures[key] = kin.context.createTexture();
            textures[key].image = new Image();
            textures[key].image.onload = function(){
                kin.initTexture(textures[key]);
                if (++loadedImages >= numImages) {
                    callback();
                }
            };
            textures[key].image.src = sources[key];
        })();
    }
}
```

La animación en el contexto 3D se hace redibujando toda la escena tantas veces como sea necesaria mientras la posición de las estructuras varía o el ángulo de la cámara o la posición del usuario dentro del contexto. La siguiente función hace que mientras la página esté abierta, ante un cambio se redibuje todo el contexto. Esta es una de las razones por la que renderizar una escena 3D es tan costoso.

```
function startAnimation(){
    kin.setDrawStage(function(){
        drawStage();
    });
    kin.startAnimation();
}
```

Las funciones restantes son los handlers de cada evento. El usuario puede moverse ya sea con el movimiento del mouse, o con las teclas. Dentro de un dispositivo táctil como ya aclaramos anteriormente la librería kinetic nos convierte los gestos táctiles a eventos de mouse, tambien seria posible agregar botones a la interface y que estos llamen a las funciones de keyDown. Estas funciones lo que hacen es modificar los parámetros de la variable *camara* que es la que contiene el ángulo de visión y posición del usuario.

```
function handleMouseDown(){
    mouseDownPos = kin.getMousePos();
    mouseDownPitch = camera.pitch;
    mouseDownYaw = camera.yaw;
}

function handleMouseMove(){
    if (mouseDownPos !== null) {
        var mousePos = kin.getMousePos();

        // update pitch
        var yDiff = mousePos.y - mouseDownPos.y;
        camera.pitch = mouseDownPitch + yDiff / kin.getCanvas().height;

        // update yaw
        var xDiff = mousePos.x - mouseDownPos.x;
        camera.yaw = mouseDownYaw + xDiff / kin.getCanvas().width;
    }
}

function handleMouseUp(){
    mouseDownPos = null;
}
```

```
function handleKeyDown(evt){
    keycode = ((evt.which) || (evt.keyCode));
    switch (keycode) {
        case 37: // left key
            sideMovement = LEFT;
            break;
        case 38: // up key
            straightMovement = FORWARD;
            break;
        case 39: // right key
            sideMovement = RIGHT;
            break;
        case 40: // down key
            straightMovement = BACKWARD;
            break;
    }
}

function handleKeyUp(evt){
    keycode = ((evt.which) || (evt.keyCode));
    switch (keycode) {
        case 37: // left key
            sideMovement = STILL;
            break;
        case 38: // up key
            straightMovement = STILL;
            break;
        case 39: // right key
            sideMovement = STILL;
            break;
        case 40: // down key
            straightMovement = STILL;
            break;
    }
}
```

Capturas del prototipo 3D

Ya vimos que es lo que hace el código del prototipo 3D, ahora veremos por medio de capturas de pantalla como funciona y el resultado obtenido. Recordaremos que en el prototipo hay definidas 4 estructuras con parámetros fijos,

y que con solo modificar esa sección sería posible crear todas las estructuras de la facultad.

Se puede ver en las siguientes capturas de pantalla, el prototipo 3D funcionando. Se ven las 4 estructuras creadas, la estructura contenedora, las texturas utilizadas para cada una, y el cambio de perspectiva de la cámara.

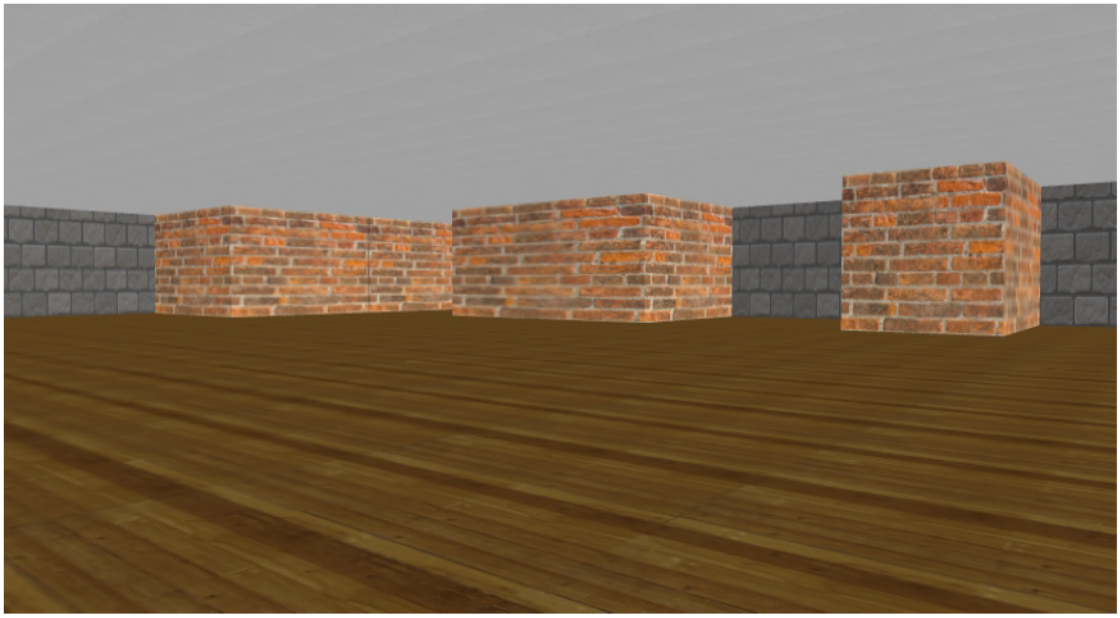


Figura 42. Ejemplo 1. Captura de la navegacion 3D

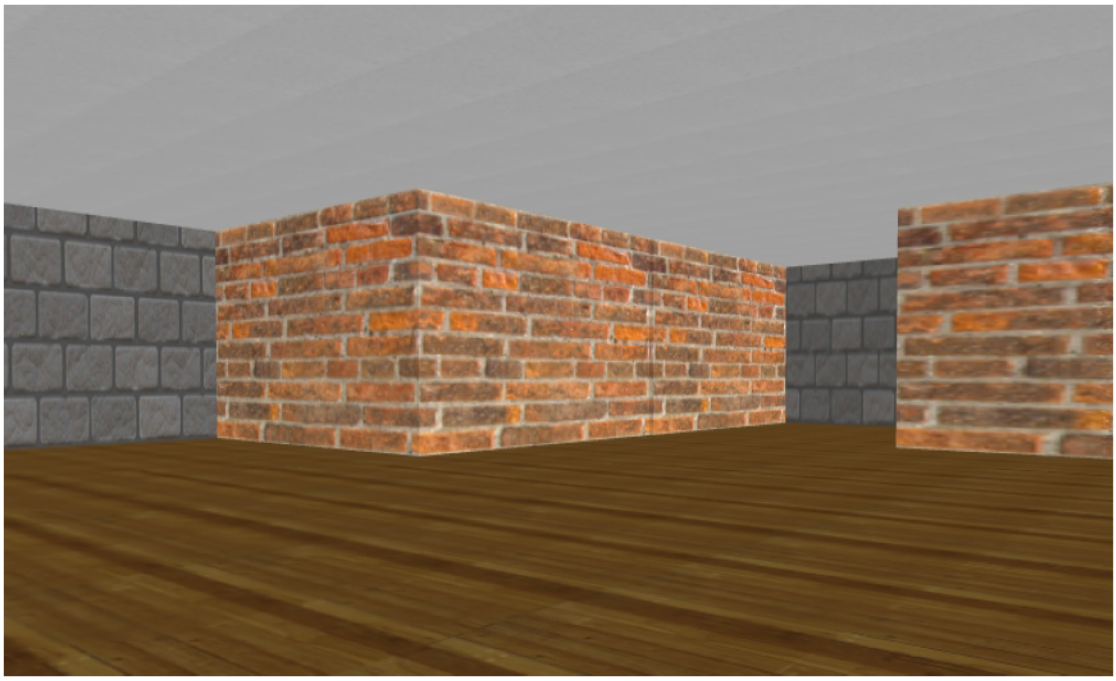


Figura 43. Ejemplo 2. Captura de la navegacion 3D

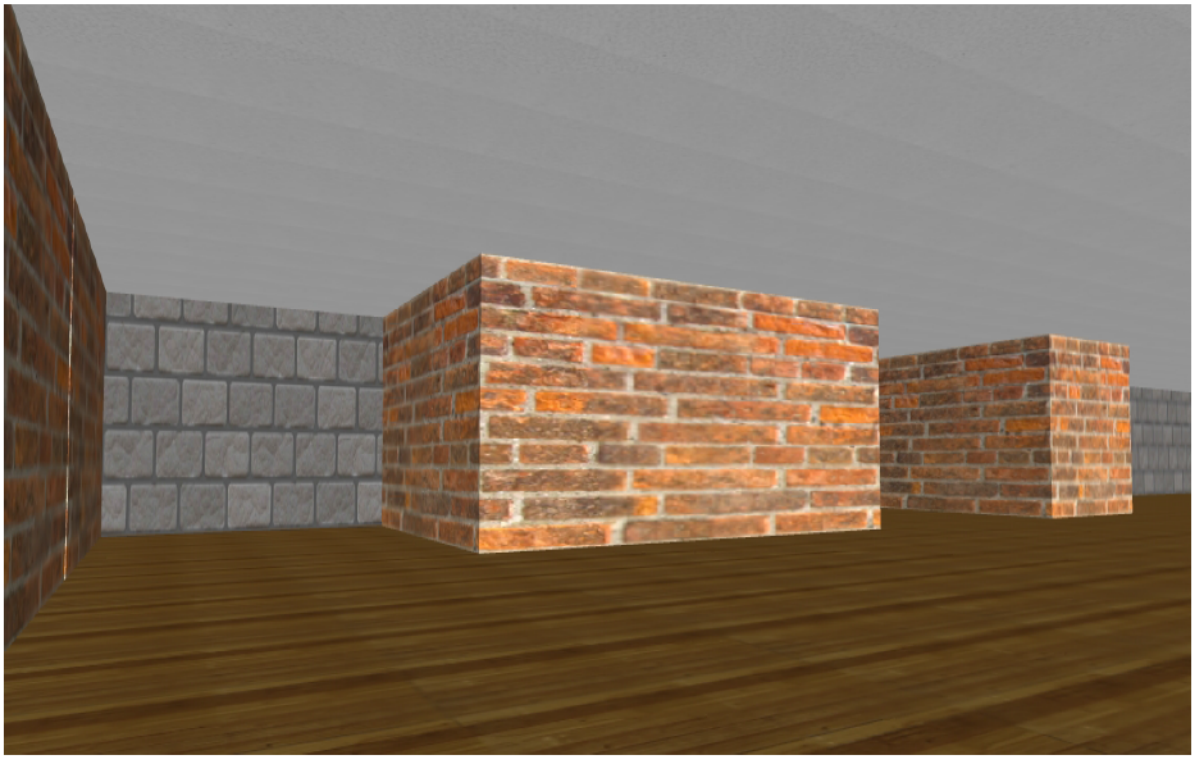


Figura 44. Ejemplo 3. Captura de la navegacion 3D

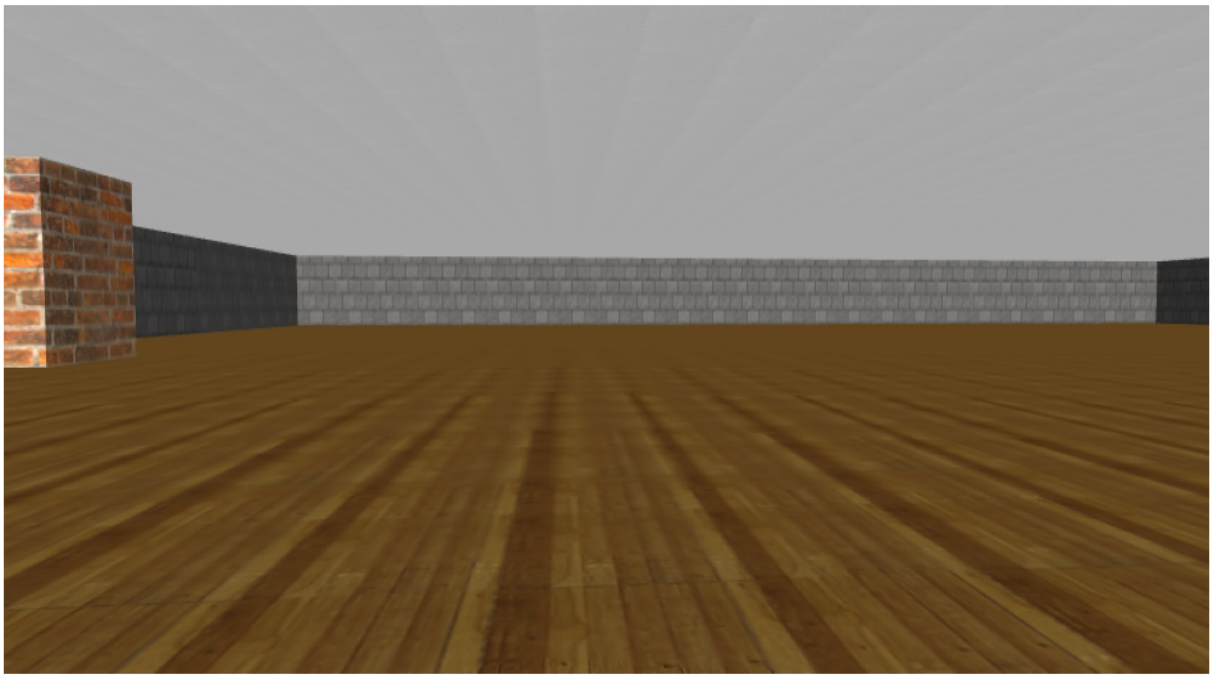


Figura 45. Ejemplo 4. Captura de la navegacion 3D

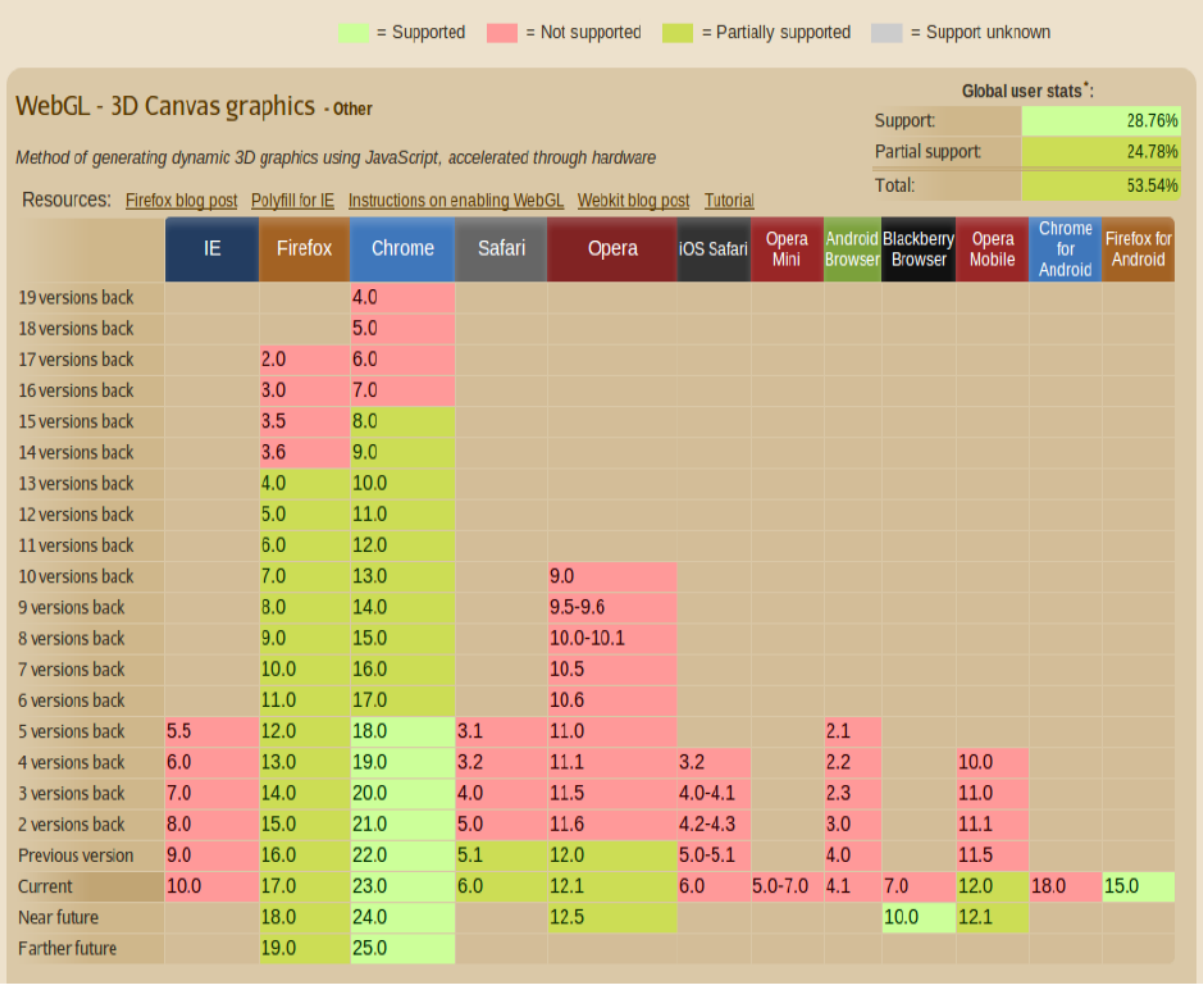
Conclusiones de la implementación 3D

La implementación de la interface 3D fue uno de los puntos que más esfuerzo y tiempo y tiempo requirió.

La complejidad de crear objetos en tres dimensiones, definiendo sus propiedades mediante matrices y transformaciones matemáticas para moverlas y ubicarlas en la posición deseada fue otro gran obstáculo.

Cuando comenzamos a investigar sobre WebGL, todo era muy nuevo, surgían nuevas librerías y nuevas versiones de las existentes, y la compatibilidad de los navegadores era mínima. Durante el desarrollo hemos visto a estas librerías evolucionar y a otras tantas desaparecer.

En el siguiente cuadro podemos ver el grado compatibilidad con WebGL de los principales browsers actuales.



= Supported = Not supported = Partially supported = Support unknown

WebGL - 3D Canvas graphics - Other

Method of generating dynamic 3D graphics using JavaScript, accelerated through hardware

Resources: Firefox blog post | Polyfill for IE | Instructions on enabling WebGL | Webkit blog post | Tutorial

Global user stats*:

Support:	28.76%
Partial support:	24.78%
Total:	53.54%

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android
19 versions back			4.0									
18 versions back			5.0									
17 versions back		2.0	6.0									
16 versions back		3.0	7.0									
15 versions back		3.5	8.0									
14 versions back		3.6	9.0									
13 versions back		4.0	10.0									
12 versions back		5.0	11.0									
11 versions back		6.0	12.0									
10 versions back		7.0	13.0		9.0							
9 versions back		8.0	14.0		9.5-9.6							
8 versions back		9.0	15.0		10.0-10.1							
7 versions back		10.0	16.0		10.5							
6 versions back		11.0	17.0		10.6							
5 versions back	5.5	12.0	18.0	3.1	11.0			2.1				
4 versions back	6.0	13.0	19.0	3.2	11.1	3.2		2.2		10.0		
3 versions back	7.0	14.0	20.0	4.0	11.5	4.0-4.1		2.3		11.0		
2 versions back	8.0	15.0	21.0	5.0	11.6	4.2-4.3		3.0		11.1		
Previous version	9.0	16.0	22.0	5.1	12.0	5.0-5.1		4.0		11.5		
Current	10.0	17.0	23.0	6.0	12.1	6.0	5.0-7.0	4.1	7.0	12.0	18.0	15.0
Near future		18.0	24.0		12.5				10.0	12.1		
Farther future		19.0	25.0									

Figura 46. Tabla comparativa de compatibilidad entre browsers y webGL[13]

[13] http://caniuse.com/webgl

Como podemos ver los navegadores con mayor compatibilidad con WebGl son los de PC, los navegadores de los dispositivos móviles como iOS safari, Opera Mini, Chrome y el navegador por defecto de Android no son compatibles aún. Esta información está basada en test y versiones de navegadores de octubre del 2012.

Cuando se implementó la interface 3D, apenas Chrome llegaba a ejecutar WebGL, los navegadores de dispositivos móviles recién estaban empezando a cubrir HTML5. Como se pudo ver en la explicación de la implementación el contexto webGL era obtenido utilizando la instrucción *getContext("experimental-webgl")* esto fué porque el contexto webgl todavia no existia.

Por estos motivos es que el prototipo 3D quedó inconcluso, el cambio del modelo de datos hizo imposible adaptar lo ya implementado al nuevo modelo de datos, donde las estructuras no son definidas por un alto y ancho, sino por un conjunto de puntos en el plano que determinan su forma.

9. Conclusiones y trabajo futuro

Daremos un repaso general del proyecto, presentando nuestras conclusiones finales en función de los resultados obtenidos y señalaremos posibles trabajos futuros que puedan presentarse como consecuencia de éste.

9.1 Conclusión

La localización en interiores en la actualidad es un problema no resuelto. Si bien hay soluciones aproximadas no son lo suficientemente precisas como muchas veces se requiere.

A lo largo de la presente tesis, se han analizado diversos elementos sobre la localización en interiores. El estudio de los mismos nos ha develado la complejidad de este tipo de sistemas debido fundamentalmente a dos factores. En primer lugar la brecha tecnológica que existe todavía en este tipo de ambientes, para mejorar la precisión, es necesario aumentar la infraestructura o bien, si se utiliza la ya existente, realizar estudios para poder satisfacer la navegación en este tipo de ambientes. En Segundo lugar, la gran mayoría de los dispositivos móviles existentes no cumplen con los requerimientos técnicos necesarios para dar soporte a las herramientas de desarrollo actuales.

Estos dos elementos de complejidad han sido los retos que esta tesis ha abordado mediante el estudio de las distintas herramientas tanto tecnológicas como de desarrollo. Como hemos podido estudiar a lo largo de los Capítulos 2, 3 y 4; existen innumerables técnicas y algoritmos de posicionamiento de las cuales no estamos seguros de tener un 100 por cien por ciento en lo que a precisión respecta. Por otro lado, la curva de desarrollo de las herramientas actuales para realizar este tipo de aplicaciones evoluciona constantemente, haciendo que ciertas librerías dejen de ser mantenidas o bien que las mismas sean actualizadas constantemente haciendo que la implementación sea muy dificultosa.

En primer lugar se diseñó e implementó una interfaz 2D que representa la planta baja del edificio de la facultad de informática sito en calle 50 y 120 de la Ciudad de La Plata.

La aplicación permite realizar lo siguiente:

- Ubicar a los usuarios en la Facultad a través de la lectura de códigos QR.
- Seleccionar el lugar a donde desean dirigirse, pudiendo volver en todo momento a la selección de otro destino.

- Durante la navegación, el usuario puede visualizar galerías multimedia de las estructuras navegables con solo hacer click sobre las mismas. También puede realizar Zoom (IN y OUT) sobre el mapa 2D.

- El prototipo simula la lectura de códigos QR permitiendo al usuario avanzar y retroceder dentro del mapa.

Todas estas funcionalidades permite a los usuarios “ubicarse” en el entorno y ser asistidos en el momento de querer llegar a una oficina o aula de la Facultad, con la ventaja de tener una interfaz que permite ver los lugares navegables, dentro del edificio.

En este punto, el mayor problema con el que nos encontrados fue el cambio que hemos tenido que realizar en el modelo de objetos planteado para la primera instancia del prototipo, ya que el mismo sólo permitía representar estructuras rectangulares.

Para la segunda instancia del prototipo se reformuló el modelo de datos permitiendo que represente estructuras irregulares (como las que presenta el edificio de la nueva facultad de informática). Para lo que fue necesario realizar un nuevo relevamiento de todas las estructuras de la facultad para así poder representar el edificio; también fue necesario el rediseño de la interfaz.

Por otra parte se implementó una interfaz 3D, la cual no ha podido ser desarrollada completamente debido a que la gran mayoría de los browsers existentes para dispositivos móviles no soportan las directivas 3D necesarias (WEBGL).

En particular, , muchas de las herramientas para crear objetos en tres dimensiones en el canvas de HTML5 son muy nuevas, muchas con fallas en la implementación, poca documentación e información sobre las mismas. La mayoría de estas APIs o Frameworks son desarrolladas hasta solucionar un problema determinado y no evolucionan mas, quedando bugs sin resolver y problemas de compatibilidad.

El contexto 3D del elemento canvas de HTML5 si bien nos permitió instanciar figuras tridimensionales, la curva de aprendizaje fue muy elevada. Definir figuras en un lenguaje de bajo nivel, declarando matrices y transformaciones matemáticas fue todo un desafío.

Cuando pudimos crear estructuras en 3D, el código resultante fue siempre demasiado para la velocidad de los procesadores de los dispositivos móviles, como ya vimos WebGL hace uso de la aceleración por hardware y no todos los dispositivos móviles cuentan con una tarjeta gráfica dedicada.

Por último los navegadores móviles no tenían soporte completo para las funciones 3D de HTML5 , cuando desarrollamos la interfaz 3D solo Google Chrome y Firefox soportaban el contexto “experimental-webgl”, los navegadores móviles estaban mucho más relegados en este aspecto.

9.2 Trabajos futuros

A continuación, se detallan ciertas funcionalidades que creemos que serían importantes para mejorar el prototipo.

Cambiar la tecnología de Códigos QR por otra que sea capaz de posicionar al usuario en el mapa sin la necesidad de tener que decirle a la aplicación donde se encuentra a cada momento.

Mejorar la herramienta de construcción de planos para que en lugar de marcar cada posición a mano, se pueda hacer algo mas automatizado, como, por ejemplo, superponer el plano y marcar puntos dando una referencia de origen.

Investigar sobre el uso del contexto 3D para implementar la posibilidad de brindarle al usuario la navegación en tres dimensiones si el dispositivo lo permite.

10. Referencias Bibliográficas

[1] Rolf A. de By, Principles of Geographics Information Systems. [en línea] <http://www.gdmc.nl/oosterom/PoGISHyperlinked.pdf>.
[2] Google Maps. [en línea] <https://maps.google.com.ar/>.
[3] Google Earth. [en línea] <http://www.google.com/earth/index.html>
[4] Bing Maps. [en línea] <http://www.bing.com/maps/>
[5] Street View. [en línea] <http://maps.google.com/intl/en/help/maps/streetview/>
[6] J.C. Torres, Diseño Asistido por Ordenador. [en línea] <http://lsi.ugr.es/~jctorres/cad/docs/resumenTema1.pdf>
[7] Julio Cesar Osma Gamboa. Esap Informática II. Sistema de Información Geográfica. [en línea] <http://www.slideshare.net/samirdejesus/sistema-de-informacin-geogrfica-15200881>
[8] Yari Xg, Fundamentos de SIG [en linea] <http://tecamb-sig.blogspot.com.ar/2012/12/primera-unidad.html>
[9] Latitud 92, ¿Que es un SIG?. [en línea] <http://www.latitude92.com/es/ventajas-esp/que-es-un-sig-esp>
[10] ¿Qué son los servicios basados en localización?. [en línea] <http://lbspro.com/?q=que-son-servicios-localizacion-LBS>
[11] Terje Midtbø, Alexander S. Nossum, Are indoor positioning systems mature for cartographic tasks?. 2012. [en línea] <http://www.cartogis.org/docs/proceedings/2012/Midtbo_etal_AutoCarto2012.pdf>
[12] Local Positioning Systems. LBS Applications and Services. Krzysztof Kolozdziej and Johan Hjelm. Ed. Taylor & Francis. Capítulo 5. Publicado en 2006 por CRC Press. Taylor & Francis Group.
[13] Jeffrey Hightower y Gaetano Borrielo, Location Systems for Ubiquitous Computing. 2001. [en línea] <http://www.intel-research.net/Publications/Seattle/062120021154_45.pdf>
[14] Hugo García Gonzáles, Sistemas de Localización Basados en TOA. [en línea] <http://www.slideshare.net/guest3014af17/sistemas-de-localizacin-basados-en-toa>
[15] Antonio R. Jiménez, Tecnologías sensoriales de localización para entornos inteligentes. [en línea] - <http://www.car.upm-csic.es/lopsi/static/publicaciones/Congreso%20Nacional/ARJimenez_SurveyLocalizacionCUbicua_CEDI_UCAmI_2005.pdf>
[16] Chip nVidia Tegra 3. [en línea] <http://www.nvidia.com/object/tegra-3-processor.html>
[17] Windows 8. [en línea] <http://windows.microsoft.com/es-xl/windows-8/meet>
[18] Forman, G y Zaharjon, J. The challenges of mobile computing. [en línea] <http://www.cs.colorado.edu/~rhan/CSCI_7143_002_Fall_2001/Papers/Forman94_Challenges.pdf>
[19] Realidad Aumentada y Educación. [en línea]. <http://www.infotecarios.com/node/174>
[20] Java Micro Edition. [en línea]. <http://www.oracle.com/technetwork/java/javame/index.html>

[21] Windows Phone. [en línea]. <http://www.windowsphone.com/es-ar>
[22] B.R.E.W. [en línea]. <https://www.brewmp.com/>
[23] Blackberry OS. [en línea]. <http://www.blackberryos.com/>
[24] Palm OS. [en línea]. <http://www.openwebosproject.org/>
[25] Iphone. [en línea]. <http://www.apple.com/es/ios/>
[26] Symbian OS. [en línea]. <http://licensing.symbian.org/>
[27] Android. [en línea]. <http://www.android.com/>
[28] Openmoko. [en línea]. <http://www.openmoco.org/>
[29] MeeGo. [en línea]. <hhttps://www.meego.com/>
[30] Limo. [en línea]. <http://www.limofoundation.org/>
[31] Aplicaciones Web vs Nativas vs Híbridas. [en línea].
<http://geospatialtraininges.com/recursos-gratuitos/tipos-de-aplicaciones-moviles/>
[32] Miguel Angel Alvarez. HTML 5. [en línea]
<http://www.desarrolloweb.com/articulos/nuevas-etiquetas-html5.html>
[33] Introducción a Jquery Mobile. [en línea].
<http://wwwhatsnew.com/2011/03/31/introduccion-a-jquery-mobile/>
[34] KineticJS. [en línea]. <https://github.com/ericdrowell/KineticJS/wiki>
[35] Tan Jin Soon, QR Code. [en línea].
<http://www.itsc.org.sg/pdf/synthesis08/Three_QR_Code.pdf>
[36] Carlos Samuel Cantero Gonzalez, Web-based Graphics Library. [en línea].
<http://books.openlibra.com/pdf/WebGL.pdf>
[37] Symfony at a Glance. [en línea]. <http://symfony.com/symfony-at-a-glance>

Otras fuentes de lectura

A. Butz, J. Baus, A. Krüger, and M. Lohse. A hybrid indoor navigation system. In IUI2001: International Conference on Intelligent User Interfaces, New York, 2001. ACM.
Chang, Y.J., Tsai, S.K., Chang, Y.S.,Wang, T.Y.: A NovelWayfinding System Based on Geo-Coded QR-Codes for Individuals With Cognitive Impairments. In: Assets '07: Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility, New York, NY, USA, ACM (2007) 231–232
E. W. Dijkstra. A note on two problems in connection with graphs. Numerische Mathematik, (51):161–166, 1950.
Gartner, G., Frank, A., Retscher, G.: Pedestrian navigation system in mixed indoor outdoor environment the navio project (2004)
Fraunhofer. Messe Navigator.
<http://www.iis.fraunhofer.de/ec/navigation/indoor/projekte/messe/index_d.html>, accessed at April 2011
Ran, L., Helal, S., Moore, S.: Drishti: An integrated indoor/outdoor blind navigation system and service. In: PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04), Washington, DC, USA, IEEE Computer Society (2004)

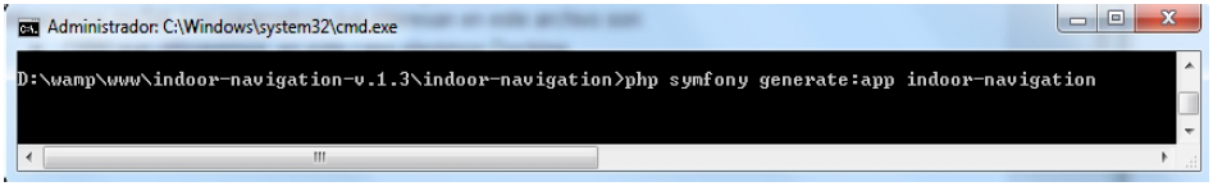
M. Sorrows and S. Hirtle. The Nature of Landmarks for Real and Electronic Spaces. In C. Freksa and D. M. Mark, editors, COSIT ’99, volume 1661, pages 37–50. Springer, 1999
Streeter, L.A., Vitello, D., Wonsiewicz, S.A.: How to Tell People Where to Go: Comparing Navigational Aids. International Journal of Man-Machine Studies 22(5) (1985) 549–562
Tsetsos, V., Anagnostopoulos, C., Kikiras, P., Hasiotis, P., Hadjiefthymiades, S.: A Human-Centered Semantic Navigation System for Indoor Environments. Pervasive Services, 2005. ICPS ’05. Proceedings. International Conference on (July 2005) 146–155

Apéndice A – Código 2D

En este capítulo se presenta como se fue desarrollando la aplicación 2D

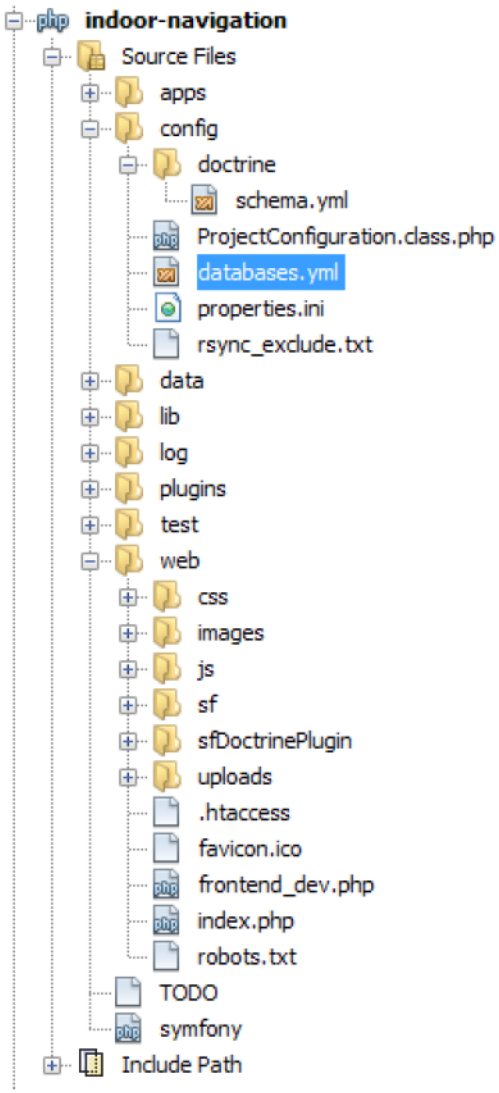
Implementación del Backend

Comenzamos utilizando la consola del sistema operativo para generar la aplicación con symfony.



"**php symfony generate:app indoor-navigation**"

Importamos el proyecto al IDE Netbeans, donde podemos ver la estructura del proyecto ya generada:



Definimos en el archivo **databases.yml** la configuración de la base de datos donde se utilizó MySQL. Los parámetros que interesan en este archivo son:

- Mapeador de objetos a la base de datos (ORM), en este caso elegimos **Doctrine**.
- El nombre de la base de datos en MySQL: indoor_navigation
- Usuario y password de la base de datos.

Utilizamos Doctrine como ORM de Symfony, y comenzamos definiendo la estructura de las entidades y sus relaciones en el archivo **schema.yml**.
Se describe en este archivo cada una de las entidades, sus propiedades, tipo de datos y las relaciones entre ellas.

Schema.yml

Definición de la clase Estructura

```yaml
#Modelo de datos Indoor Navigation 2D/3D
Estructura:
  columns:
    nombre:
      type:     string(50)
      notnull:  true
    tipo:
      type: integer
      notnull: true
    capacidad:
      type:     integer
      notnull:  true
    es_navegable:
      type:     boolean
      default:  true
```

Definición de la clase Multimedia

```yaml
Multimedia:
  columns:
    nombre:
      type:     string(255)
      notnull:  true
    tipo:
      type: integer
      notnull: true
    estructura_id:
      type:     integer
  relations:
    Estructura:
      refClass:   Estructura
      onDelete:   cascade
      local:      estructura_id
      foreign:    id
```

Definición de la clase Puntos

```
34  Puntos:
35    columns:
36      punto_origen_x:
37        type:     integer
38        notnull:  true
39      punto_origen_y:
40        type:     integer
41        notnull:  true
42      estructura_id:
43        type:     integer
44    relations:
45      Estructura:
46        refClass:   Estructura
47        onDelete:   cascade
48        local:      estructura_id
49        foreign:    id
50
```

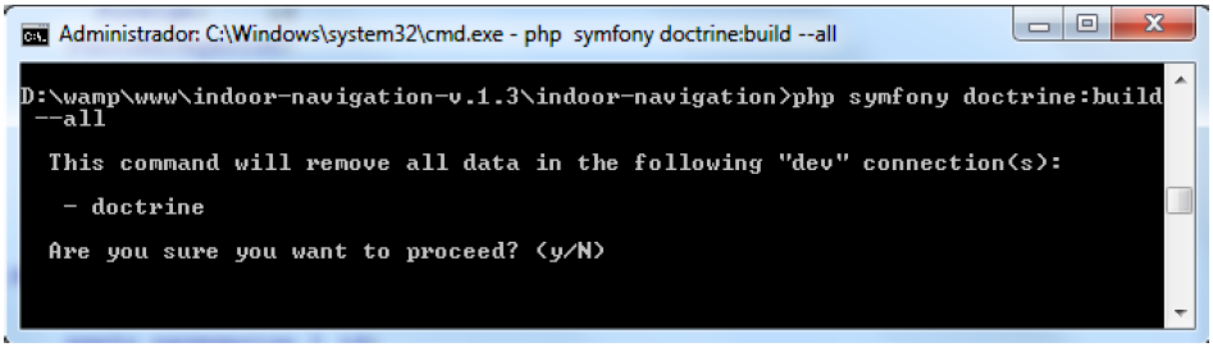
Definición de la clase PuntoNavegacion

```
51  #-----------------------------------------------
52  #    modelo que define la union de puntos     #
53  #-----------------------------------------------
54  PuntoNavegacion:
55    columns:
56      nombre:
57        type:     string(50)
58        notnull:  true
59      punto_origen_x:
60        type:     integer
61        notnull:  true
62      punto_origen_y:
63        type:     integer
64        notnull:  true
65      estructura_id:
66        type:     integer
67    relations:
68      Estructura:
69        refClass:   Estructura
70        onDelete:   restrict
71        local:      estructura_id
72        foreign:    id
73      PuntoNavegacion:
74        class: PuntoNavegacion
75        local: punto_navegacion_1_id
76        foreign: punto_navegacion_2_id
77        refClass: PuntoNavegacionPuntoNavegacion
78        equal: true
79
```

Definición de la clase PuntoNavegacionPuntoNavegacion

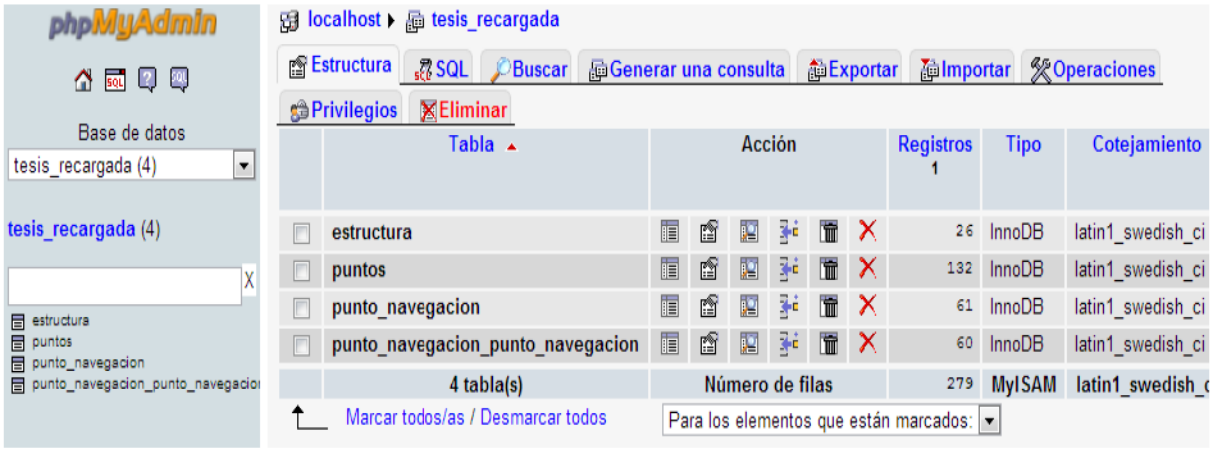
```
80  PuntoNavegacionPuntoNavegacion:
81    columns:
82      punto_navegacion_1_id:
83        type:     integer
84        primary: true
85      punto_navegacion_2_id:
86        type:     integer
87        primary: true
88      distancia:
89        type:     float
90        notnull:  true
```

Una vez definida la estructura de las entidades, por consola indicamos que cree las clases base, y haga el mapeo a la base de datos definida en el archivo **databases.yml**.

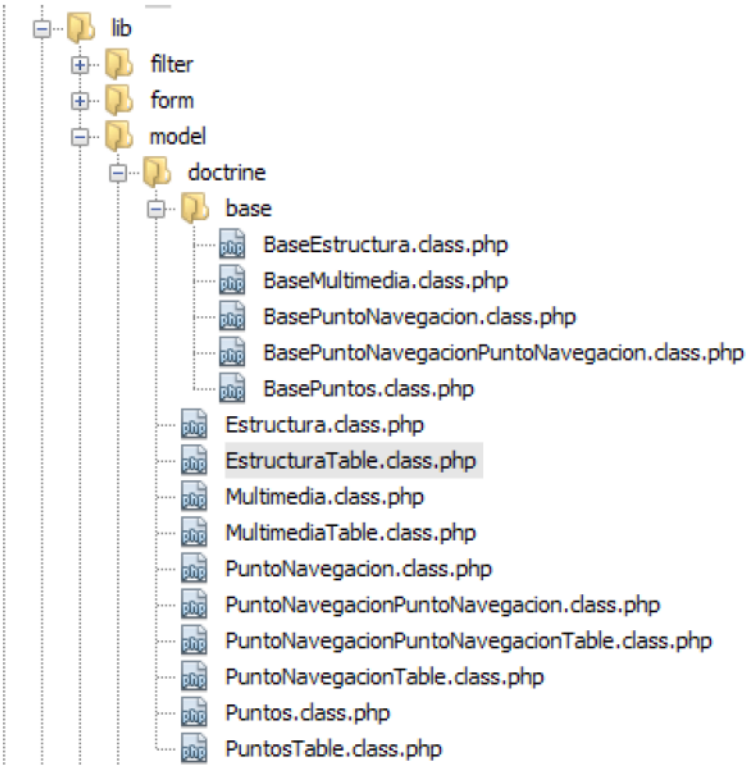


"**php symfony doctrine:build --all**"

podemos ver como a través del último comando, quedan creadas las tablas en la base de datos.



Symfony genera clases a partir de la estructura definida en el **schema.yml**. Clases Base (con todos los métodos necesarios para manejar los atributos de cada clase), y subclases para realizar los métodos necesarios.



Ya tenemos las clases necesarias para guardar los datos de cada estructura y los puntos de navegacion, con su correspondiente mapeo a la base de datos MySQL.

La instanciación de las clases fueron realizadas a través de la herramienta de administración.

Indoor Navigation Administración - Pantalla de Login

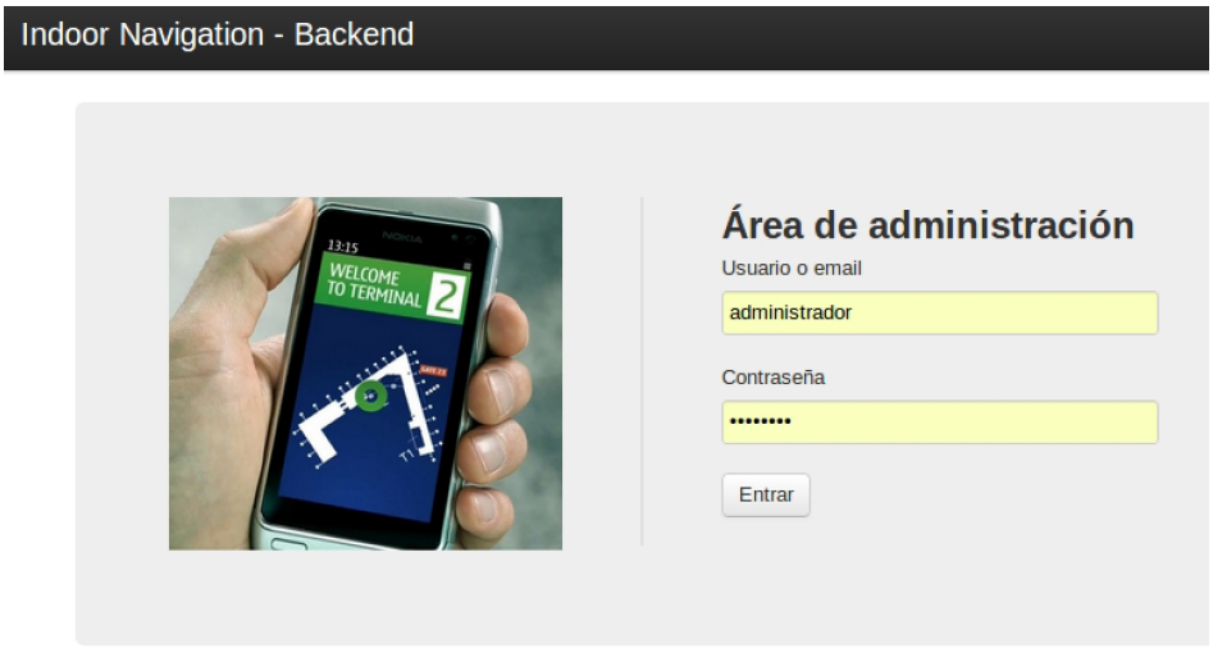


Figura 47. Pantalla aplicación Backend

Indoot Navigation - Listado de Estructuras

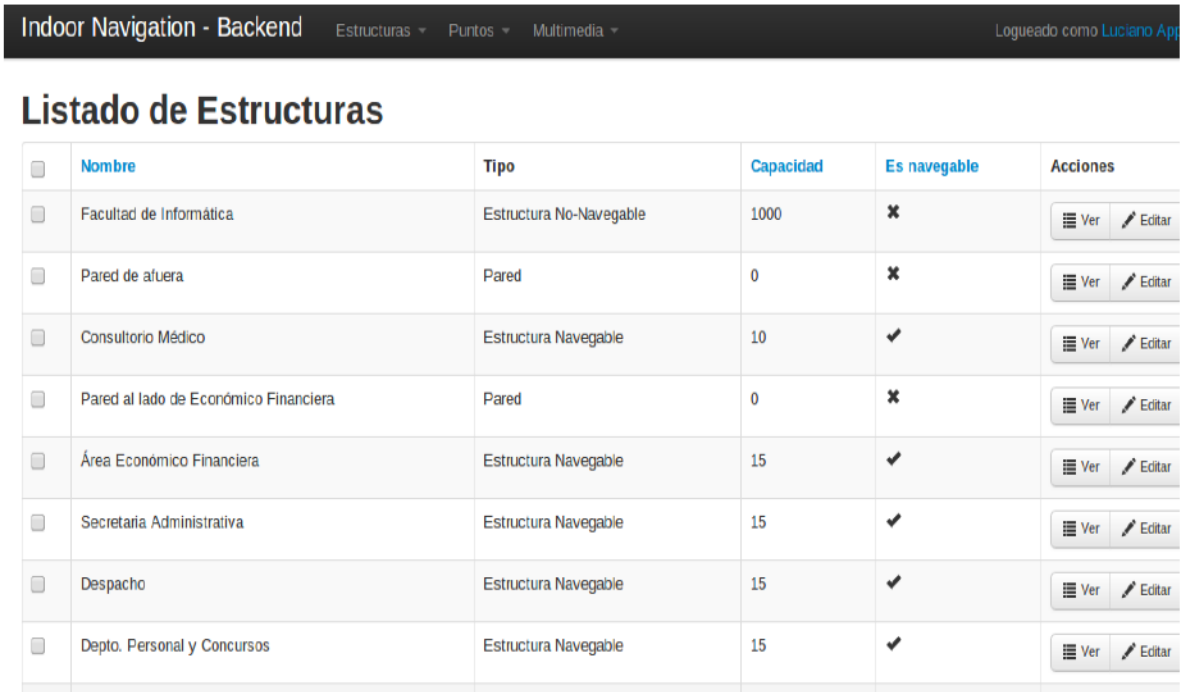


Figura 48. Pantalla aplicación Backend – Listado de estructuras

Una vez creadas las estructuras y puntos de navegación, podemos definir con estos elementos un grafo, donde los nodos son puntos de navegación y las

aristas son las relaciones entre estos puntos de navegación. Como explicamos anteriormente hay estructuras que son navegables, por lo tanto, tienen un punto de navegación asociado.
Es así cómo podemos determinar el camino de un punto actual a una estructura, solo debemos consultar el punto de navegación actual y cuando se elige el destino tendremos el segundo punto de navegación. La ruta la determinamos simplemente calculando el camino minimo entre estos dos puntos de navegacion.

- Creamos una clase que tendrá la responsabilidad de calcular el camino mínimo entre dos puntos de navegación.
- Utilizamos el algoritmo de dijkstra que se adapta perfectamente para la resolución de nuestro problema.

A continuación se pueden ver las partes más importantes de esta clase:

Definición de la clase Dijkstra y las variables necesarias

Definición de constantes

```php
<?php

class Dijkstra {

  const valor_maximo = '9999999';
  public $lista_de_adyacencias = array();
  public $cantidad_nodos;
  public $camino_a_recorrer = array();

  public $distancia = array();
  public $camino = array();

  /*
   * carga la lista de adyacencia con todos los puntos de navegacion,
   * y para cada uno agrega un arreglo.
   * Con sus vecinos y su distancia a ellos. Ejemplo.
   * 0 =>
    array
      0 =>
        array
          'punto_navegacion_1_id' => string '1' (length=1)
          'punto_navegacion_2_id' => string '2' (length=1)
          'distancia' => string '200.00' (length=6)
  */

```

Como podemos observar en la pantalla. Primero tenemos la definición de la clase y a continuación la definición de las variables necesarias para realizar el algoritmo.

- ***valor_maximo***: es una constante utilizada para la inicialización del arreglo de distancias entre 2 puntos, definiendo un valor alto para luego actualizarla con la distancia mínima entre estos dos nodos.

- ***lista_de_adyacencia***: en esta variable tendremos cada uno de los nodos y cada uno de sus vecinos.

- ***cantidad_nodos***: como el nombre de la variable lo indica, contiene la cantidad de nodos del grafo.

- ***camino_a_recorrer***: es un arreglo en que va guardando la distancia mínima entre los nodos y una vez finalizado el cálculo, se tiene todos los nodos a recorrer para llegar en la menor distancia al un destino.

- ***distancia***: en este arreglo se guarda cual es la distancia para llegar a cada uno de los nodos, son inicializados con un valor alto para que a medida que se ejecuta el cálculo del camino mínimo se pueda actualizar con el valor más bajo.

Funciones getCamino() y cargarListaDeAdyacencia()

```
26  public function getCamino(){
27    return $this->camino_a_recorrer;
28  }
29
30  public function cargarListaDeAdyacencia(){
31
32    $puntos_navegacion = Doctrine::getTable('PuntoNavegacion')->findAll();
33    foreach ($puntos_navegacion as $pd){
34      $this->lista_de_adyacencias[] = $pd;
35    }
36
37    for ($i=0;$i<count($this->lista_de_adyacencias);$i++){
38      $this->lista_de_adyacencias[$i] =
          PuntoNavegacionPuntoNavegacionTable::retrieveById($this->lista_de_adyacencias[$i]->getId());
40    }
41
42    $this->cantidad_nodos = count($this->lista_de_adyacencias);
43  }
44
```

Función caminoMinimoDesdeHasta()

```
58   public function caminoMinimoDesdeHasta($global_id_inicio, $id_destino){
59     $distancia = array();
60     $visitado = array();
61     $camino = array();
62
63     for ($i=0; $i < $this->cantidad_nodos ; $i++) {
64       $visitado[$i] = false;
65       $distancia[$i] = '9999999' ;//$this->valor_maximo;
66       $camino[$i] = -1;
67     }
68     $distancia[ $global_id_inicio -1 ] = 0;
69
70     while (!$this->noEstenTodosVisitados($visitado)){
71       $nodo_actual = $this->getMenorDesconocido($visitado,$distancia);
72       $visitado[$nodo_actual] = true;
73       foreach ($this->lista_de_adyacencias[$nodo_actual] as $vecinos){
74         if (($nodo_actual+1) == $vecinos['punto_navegacion_1_id'])
           $vecino_id = $vecinos['punto_navegacion_2_id'];
76       else
           $vecino_id = $vecinos['punto_navegacion_1_id'];
78
79         $vecino_id --;

80         //Actualizo la distancia si es menor a la del nodo actual mas la que tengo
81         if ((!$visitado[$vecino_id]) &&
82             ($distancia[$vecino_id] > $distancia[$nodo_actual] + $vecinos['distancia'])){
83           $distancia[$vecino_id] = $distancia[$nodo_actual] + $vecinos['distancia'];
84           $camino[$vecino_id] = $nodo_actual;
85         }
86       }//End Foreach
87     }//End While
88
89     $this->camino = $camino;
90     $this->distancia = $distancia;
91     $this->camino_a_recorrer =$this->imprimirCamino($id_destino, $camino, $distancia);
92   }
```

La función ***caminoMinimoDesdeHasta()*** es la encargada de calcular el camino mínimo. Recibe como parámetros el punto de navegación inicial y el de destino. Comienza inicializando las variables y poniendo los nodos como no visitados, con distancias grandes y el arreglo de camino en valores nulos. Luego crea un bucle que se repite mientras no sean visitados cada uno de los nodos, en el cual toma como nodo actual al nodo no visitado con menor distancia, lo marca como visitado y busca en la lista de adyacencia los nodos que tienen conexión con el nodo actual, y si la distancia entre nodos resultan menor a las que contiene el arreglo de distancias, procede a actualizar la distancia. Este bucle se repite hasta recorrer todos los nodos del grafo, y finalmente, se obtiene en el arreglo de distancias, las distancias mínimas entre nodos, y en el de camino, los ids de cada uno de los nodos a recorrer para llegar de un punto a otro.

Implementación del Controlador

Como pudimos observar en la sección anterior, tenemos en el backend todos los datos necesarios para representar el edificio de la facultad y navegarlo.
A continuación, describimos el controlador, el cual actúa como nexo entre la capa de presentación y el backend.

En Symfony se definen acciones "***actions***", las cuales reciben como parámetro el objeto "***request***" el cual contiene información sobre la capa de presentación. De esta forma en el controlador definimos acciones para hacer búsquedas de estructuras, navegar de un punto de navegacion a otro y movernos en el mapa de la facultad.

A continuación se describe la funcionalidad del controlador *actions.class.php* :

Función Pública - executeCrearEstructuras()

```
17    public function executeCrearEstructuras(sfWebRequest $request){
18      $crear_objectos = new Utilities();
19      $crear_objectos->crearObjetos();
20      $this->getUser()->setFlash('notice','Estructuras creadas satisfactoriamente.');
21      $this->redirect('@homepage');
22    }
```

La función *crearEstructuras()* nos realizar la carga de todas los objetos instanciados la clase **Utilities.php**.

Función Pública - executeBuscar()

```
54    /**
55     * Función que dibuja el template Busqueda
56     * @param sfWebRequest $request
57     */
58    public function executeBuscar(sfWebRequest $request){
59      //En actual ID se seteará el parámetro que vendrá del código de barra.
60      $this->getUser()->setAttribute('actual_id',1);
61      $this->getUser()->setAttribute('borrados', array());
62      $this->getUser()->setAttribute('escala',sfConfig::get('app_escala'));
63      $this->estructuras = EstructuraTable::getNavegables();
64    }
```

La función *buscar()* recupera de la base de datos todas las estructuras navegables para que el usuario pueda elegir el destino al cual navegar.

Función Pública - executeNavegar()

```
25      * Función que permite navegar desde un punto de la facultad (situado por una aplicación externa) a ot
26      * @param sfWebRequest $request
27      */
28     public function executeNavegar(sfWebRequest $request){
29       //Se setea en el usuario el id de la estructura a donde el usuario desea navegar
30       $this->getUser()->setAttribute('fin_id',$request->getParameter('est_id'));
31       $this->estructuras = Doctrine::getTable('Estructura')->findAll();
32       $d = new Dijkstra();
33       $d->cargarListaDeAdyacencia();
34
35       $estructura = Doctrine::getTable('Estructura')->find($request->getParameter('est_id'));
36
37       if ((!$estructura) || (!$estructura->getEsNavegable())){
38         $this->getUser()->setFlash('error','El lugar a donde intenta navegar no está disponible.');
39         $this->redirect('@homepage');
40       }
41
42       //Relative URL Ej: http://localhost/indoor-navigation
43       $this->basepath = ('http'.($request->isSecure() ? 's' : '').'://').
44                         $request->getHost().$request->getRelativeUrlRoot();
45       //Si es Producción index.php sino frontend_dev.php
46       $this->enviroment = (sfConfig::get('sf_environment') == 'dev' ? 'frontend_dev.php' : 'index.php');
47
48       $punto_navegacion = PuntoNavegacionTable::getPuntoNavegacionParaEstructuraId($estructura->getId());
49       $this->destino = $punto_navegacion->getId();
50       $d->caminoMinimoDesdeHasta($this->getUser()->getAttribute('actual_id'), $this->destino);
51       $this->puntos_navegacion = $d->getCamino();
52     }
```

La función *navegar(), realiza los siguientes pasos:*

- Toma del request los parámetros de selección del usuario en la función *buscar()* “punto de navegación actual y el ID de destino”.
- Crea una instancia de la clase *Dijkstra* para el cálculo de caminos e inicializa la lista de adyacencias.
- Valida si el punto de navegación seleccionado por el usuario se encuentra navegable.
- Utiliza la función *caminoMinimoDesdeHasta() de la clase Dijkstra con los parámetros seleccionados por el usuario,* y retorna a la capa de presentación el arreglo con el camino de puntos de navegación a dibujar.

Función Pública - executeDetalleEstructura()

```
66    /**
67     * Acción que es ejecutada por el usuario en el template navegar cuando el usuario realizar un,
68     * click sobre alguna estructura
69     * @param sfWebRequest $request
70     */
71    public function executeDetalleEstructura(sfWebRequest $request){
72      $this->destino =
73        Doctrine::getTable('PuntoNavegacion')->find($request->getParameter('idDestino'))->getEstructuraId();
74      $this->estructura =
75        Doctrine::getTable('Estructura')->find($request->getParameter('idEstructura'));
76      $this->multimedia =
77        MultimediaTable::getMultimediaPara($request->getParameter['idEstructura']);
78    }
```

La función *detalleEstructura()* toma del request los parámetros de selección del usuario “El Id de la estructura que desea ver en detalle y el Id de Destino actual”. Busca en la base de datos toda la información referente, y devuelve a la capa de presentación las variables necesarias para que el usuario pueda observar su selección

Public Function - executeZoomIn() y executeZoomOut()

```
112   /**
113    * Acción que es ejecutada por el usuario en el template navegar cuando el usuario ejecuta la acción "Zoom In"
114    * @param sfWebRequest $request
115    */
116   public function executeZoomIn(sfWebRequest $request){
117     //Se resta uno a la escala actual
118     $this->getUser()->setAttribute('escala', $this->getUser()->getAttribute('escala')-1);
119     //Se redirige al navegar con los nuevos parámetros seteados
120     $this->redirect('main/navegar?est_id='.$this->getUser()->getAttribute('fin_id'));
121   }
122
123   /**
124    * Acción que es ejecutada por el usuario en el template navegar cuando el usuario ejecuta la acción "Zoom Out"
125    * @param sfWebRequest $request
126    */
127   public function executeZoomOut(sfWebRequest $request){
128     //Se resta uno a la escala actual
129     $this->getUser()->setAttribute('escala', $this->getUser()->getAttribute('escala')+1);
130     //Se redirige al navegar con los nuevos parámetros seteados
131     $this->redirect('main/navegar?est_id='.$this->getUser()->getAttribute('fin_id'));
132   }
133 }
```

Las funciones *zoomIn()* y *zoomOut()* realizan la modificación del valor de la escala en el que se está encuentra dibujado el edificio y sus partes. Este valor es modificado cuando el usuario de la aplicación dispara la acción desde la vista.

Implementación del Frontend

Habiendo explicado cómo interactúa el controlador con el modelo, estamos listos para mostrar la capa de presentación.

Template - layout.php

```
<body>
  <div data-role="page" style="width:100%">
    <div data-role="header">
      <h1 style="font-size: 22px">Navegaci&oacute;n Indoor</h1>
    </div><!-- /header -->

    <div data-role="navbar">
        <ul>
          <li><?php echo link_to('Buscar', 'main/buscar', array("rel" => "external", "style" => "font-size: 20px"));?></li>
          <li><?php echo link_to('Ayuda', 'main/ayuda', array("style" => "font-size: 20px")); ?></li>
        </ul>
    </div><!-- /navbar -->

    <div data-role="content">
      <?php if ($sf_request->getParameter('action') == 'navegar'):?>
        <?php include_partial('global/barra_navegacion');?>
      <?php endif;?>
      <?php include_partial('global/mensajes_usuario');?>
      <div id="container"> </div><!--/container - Acá se dibuja el canvas-->
      <?php echo $sf_content; ?>
    </div><!-- /content -->

    <div data-role="footer">
      <h4 style="font-size: 22px"> &copy; 2012 | Navegaci&oacute;n Indoor 2D | Versi&oacute;n <?php echo sfConfig::get('app_
    </div><!-- /footer -->
  </div><!-- /div data-role -->
</body>
```

Este script PHP es el encargado de estructurar la página principal de la aplicación frontend. Cada uno de los <divs> posee un atributo (data-role) que nos permite clasificar el tag HTML para darle ciertas propiedades (a través del framework **jQueryMobile**.)

***A continuación, se detallan las propiedades que estructuran la página*:**

** **: Por defecto este atributo nos define la estructura de la página

** **: Bloque cabecera de la página.

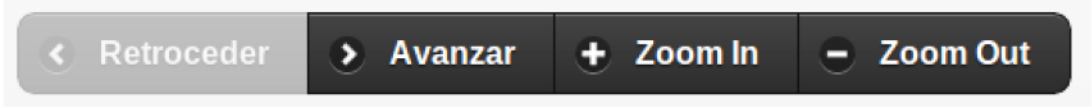
** **: Bloque a utilizar para la barra de navegación

** **: Bloque que se utiliza para mostrar el contenido dinámico de la página. En este bloque veremos el resultado de disparar las acciones *buscar()*, *navegar()*, *detalleEstructura(), etc.*

Template - barra_de_navegacion.php

```
<?php
    //Relative URL Ej: http://localhost/indoor-navigation
    $basepath = ('http'.($sf_request->isSecure() ? 's' : '').'://').$sf_request->getHost().$sf_request->getRelativeUrlRoot();
    //Si es Producción index.php sino frontend_dev.php
    $enviroment = (sfConfig::get('sf_environment') == 'dev' ? 'frontend_dev.php' : 'index.php');
;?>

<div data-role="controlgroup" data-type="horizontal" data-mini="false">
  <a id="link_retroceder" href="<?php echo $basepath."/".$enviroment."/main/retroceder";?>"
     data-role="button" data-icon="arrow-l" data-theme="a" rel = "external"> Retroceder
  </a>
  <a id="link_avanzar" href="<?php echo $basepath."/".$enviroment."/main/adelantar";?>"
     data-role="button" data-icon="arrow-r" data-theme="a" rel = "external"> Avanzar
  </a>
  <a id="link_zoom_in" href="<?php echo $basepath."/".$enviroment."/main/zoomIn";?>"
     data-role="button" data-icon="plus" data-theme="a" rel = "external"> Zoom In
  </a>
  <a id="link_zoom_out" href="<?php echo $basepath."/".$enviroment."/main/zoomOut";?>"
     data-role="button" data-icon="minus" data-theme="a" rel = "external"> Zoom Out
  </a>
</div>
```



La barra de navegación le permite al usuario realizar diferentes acciones sobre el mapa que se dibuja en el canvas.

** **: Bloque de pie de página

Template - buscarSuccess.php

```
<?php use_helper("I18N");?>
<h3 style="font-size: 20px"><?php echo __('Seleccione el destino.');?></h3>
<ul data-role="listview" data-inset="true" data-filter="true" data-filter-placeholder="Escriba parte del nombre para fil
  <?php foreach ($estructuras as $estructura):?>
  <li> <?php echo link_to($estructura->getNombre(),'main/navegar?est_id='.$estructura->getId(),array(
              "rel" => "external", "style" => "font-size: 18px"));?>
  </li>
  <?php endforeach?>
</ul>
```

En el template *buscar,* se recorren todas las estructuras y para cada una dibuja se dibuja un tag <a> con el nombre de la estructura. Como podemos observar, el tag <ul> tiene ciertos atributos propios del framework jQueryMobile. Permitiéndonos mostrar los items en forma de lista con un buscador.

Figura 49. Pantalla de Busqueda aplicación Indoor-Navigation

Al hacer click sobre el nombre de una estructura, será disparada la acción ***navegar()*** enviando como parámetro el **ID** de la estructura seleccionada.

Template - navegarSuccess.php

Este template es el encargado de dibujar en el canvas el plano completo de la facultad y el camino hacia la estructura seleccionada por el usuario, a continuación detallamos en profundidad el código del template.

```
<script>
  $(document).ready(inicio);
    var stage = new Kinetic.Stage({
          container: "container",
          width: <?php echo sfConfig::get('app_canvas_width');?>,
          height: <?php echo sfConfig::get('app_canvas_height');?>
    });

  var layer = new Kinetic.Layer();

  function inicio(){
    dibujarFacu(stage);
    dibujarTodosLosPuntos(stage,layer);
    deshabilitarBotonRetroceder();
    deshabilitarBotonAvanzar();
    deshabilitarBotonZoomOut();
    deshabilitarBotonZoomIn();
    //Centra el canvas cada vez que realizamos una petición
    centrar(stage);
  }
```

En el **document.ready** instanciamos nuestro canvas utilizando un método de clase que nos provee la librería de Kinetic. Le pasamos el ID “container” del elemento HTML “ ” que utilizaremos como bloque para mostrar el canvas, y un alto y un ancho (parametrizados en la configuración de la aplicación).
Como podemos observar, al document.ready se le pasa como parámetro la **función inicio(),** está función dispara todos los métodos que dibujan la facultad, las estructuras, los puntos, y la barra de navegación.

función dibujarFacu()

```
<?php use_helper("JavascriptBase");?>
<script>
  _BASEPATH = "<?php echo $basepath;?>";
  _ENVIROMENT = "<?php echo $enviroment;?>";

  function dibujarFacu(stage){
    <?php foreach ($estructuras as $estructura):?>
      <?php $puntos = PuntosTable::getPuntosBy($estructura->getId());?>
      <?php if (count($puntos) > 0):?>
        <?php $arreglo_de_puntos = array()?>
        <?php for($i = 0; $i < count($puntos)-1; $i++):?>
          <?php $arreglo_de_puntos[] = $puntos[$i]->getPuntoOrigenX()/($sf_user->getAttribute('escala'))+10;?>
          <?php $arreglo_de_puntos[] = (sfConfig::get('app_maximo_y') - $puntos[$i]->getPuntoOrigenY())
                                        / $sf_user->getAttribute('escala');?>
        <?php endfor;?>
          <?php $arreglo_de_puntos[] = $puntos[count($puntos)-1]->getPuntoOrigenX()/$sf_user->getAttribute('escala')+10;?>
          <?php $arreglo_de_puntos[] = (sfConfig::get('app_maximo_y') - $puntos[count($puntos)-1]->getPuntoOrigenY())
                                        / $sf_user->getAttribute('escala');?>
            dibujarPoligono(stage
                , "<?php echo $estructura->getTipo();?>"
                , "<?php echo $estructura->getId();?>"
                , "<?php echo $destino;?>"
                , new Array(<?php echo implode(',',$arreglo_de_puntos);?>));
      <?php else:?>
        <?php echo ('No existen estructuras cargadas para ser dibujadas');?>
      <?php endif;?>
    <?php endforeach;?>}
</script>
```

Esta función recorre todas las estructuras y para cada una de ellas realiza una consulta buscando sus puntos. Por cada uno de los puntos que tiene la estructura, se realiza una conversión de los puntos de origen (x,y) utilizando una escala seteada en la configuración de la aplicación. Por cada estructura, llamamos a la función ***dibujarPoligono()***, encargada de pintar en el canvas cada estructura.

función dibujarPoligono()

```
function dibujarPoligono(stage, tipoEstructura, idEstructura, idDestino, pointsToDraw){
  var poly = new Kinetic.Polygon( { points: pointsToDraw, fill: '#75D6B5', stroke: "black", strokeWidth: 2 });

  if (tipoEstructura == 1){
    poly.on("touchstart click", function() {
      window.location = _BASEPATH + "/" +
                        _ENVIROMENT + "/main/detalleEstructura?idEstructura="+idEstructura+"&idDestino="+idDestino;
    });

    poly.on("touchstart click", function() {
      var complexText = layer.get("#myText")[0];
      layer.remove(complexText);
    })

    poly.setFill('#75D6B5');
  }
  else if (tipoEstructura == 2){
    poly.setFill('#FFCC99');
  }
  else{
    poly.setFill('#BCA0A0');
  }

  //add the poly to the layer
  layer.add(poly);
  // add the layer to the stage
  stage.add(layer);
}
```

La función recibe como parámetro el canvas (stage), el tipo del cuál es la estructura, el id de la estructura en cuestión, el id de destino actual elegido por el usuario, y los puntos que forman la estructura. La función simplemente dibuja en el canvas cada estructura utilizando el método de clase **Poligon()** que nos provee la librería Kinetic. Este método recibe los puntos donde deseamos que el polígono se dibuje, y las propiedades de estilos de la figura.
Dependiendo del tipo de estructura (Si es navegable o no) cada polígono se dibuja con colores diferentes.

Para cada figura se genera un link que nos permite ver sus propiedades en detalle. Para no perder las referencias de la navegación actual, se envía a la acción ***detalleEstructura()*** el id de la estructura que deseamos explorar y el id de navegación actual.

función dibujarTodosLosPuntos()

```
<script>
  function dibujarTodosLosPuntos(stage,layer){
      <?php $arreglo_de_puntos = array()?>
      <?php $aux = 1;?>
      <?php $sf_user->setAttribute('siguiente_id',$puntos_navegacion[count($puntos_navegacion)-2]->getId());?>
      <?php foreach($puntos_navegacion as $punto_navegacion):?>
          <?php
            $id = $punto_navegacion->getId();
            $x = $punto_navegacion->getPuntoOrigenX() /$sf_user->getAttribute('escala') + 10;
            $y = (sfConfig::get('app_maximo_y') - $punto_navegacion->getPuntoOrigenY())
                                    / $sf_user->getAttribute('escala');
            $arreglo_de_puntos[] = $x;
            $arreglo_de_puntos[] = $y;
          ?>
          <?php if ($aux == count($puntos_navegacion)):?>
            dibujarPuntoNavegacion(stage,layer,<?php echo $x?>,<?php echo $y?>, <?php echo $id?>, true);
          <?php else:?>
            dibujarPuntoNavegacion(stage,layer,<?php echo $x?>,<?php echo $y?>, <?php echo $id?>, false);
          <?php endif;?>
          <?php $aux++;?>
      <?php endforeach;?>
      dibujarLinea(stage, layer, new Array(<?php echo implode(',',$arreglo_de_puntos);?>));
  }
</script>
```

Esta función es la encargada de dibujar todos los puntos entre la ubicación actual del usuario y la estructura seleccionada en la navegación (destino). Se setea como atributo del usuario el id del siguiente punto a visitar, pudiendo de esta manera emular los pasos que iría realizando el usuario en el mapa.

función dibujarPuntoNavegacion()

```
function dibujarPuntoNavegacion(stage,layer,x,y,id, primero){
  // Si primero, se carga una imagen en vez de dibujar un punto
  if(primero){
    var imageObj = new Image();
    imageObj.onload = function() {
      var imageUser = new Kinetic.Image({
        x: x-8, y:
        y-8, image: imageObj,
        width: 16,
        height: 16 });
        layer.add(imageUser);
        stage.add(layer);
    }
    imageObj.src = _BASEPATH + "/images/people1.png";
   }//Si no es el punto inicial dibujamos un punto de navegacion simple
   else{
     var circle = new Kinetic.Circle({
       x: x,
       y: y,
       radius: 2,
       fill: "#1C777B",
       stroke: "black",
       strokeWidth: 4 });

     layer.add(circle);
     stage.add(layer);
   }
}
```

La función ***dibujarPuntoNavegación()*** es llamada por ***dibujarTodosLosPuntos()***, tantas veces como cantidad de puntos serán dibujados. Por cada llamado la función dibuja un punto de navegación utilizando el método de clase **Circle()** de la librería Kinetic, este método recibe como parámetro las coordenadas (X,Y) donde queremos que el punto se dibuje en el canvas y ciertos atributos de estilos.

Si la función ***dibujarPuntoNavegación()*** recibe como parámetro primero en true, en vez de dibujar un punto pondrá una imagen (representación del lugar donde se encuentra parado el usuario).

función dibujarLinea()

```
function dibujarLinea(stage, layer, pointsToDraw){
  var lineaCamino = new Kinetic.Line({
          points: pointsToDraw,
          stroke: "green",
          strokeWidth: 2,
          lineJoin: "round",
          dashArray: [5, 2]
        });
  layer.add(lineaCamino);
  stage.add(layer);
}
```

La función ***dibujarLinea()*** es llamada por ***dibujarTodosLosPuntos()***. Simplemente realiza una línea de trazos entre la ubicación actual del usuario y la estructura seleccionada en la navegación (destino). Para realizar la unión entre los puntos, se utiliza el método de clase **Line()** que nos provee Kinetic, el cuál recibe como parámetro los puntos donde queremos dibujar una línea y ciertos atributos de estilos.

función deshabilitarBotonRetroceder()

```
function desabilitarBotonRetroceder(){
  <?php if (count($sf_user->getAttribute('borrados')) == 0 ):?>
    $('#link_retroceder').addClass('ui-disabled');
  <?php endif;?>
}
```

Función que deshabilita el botón retroceder de la barra de navegación dependiendo de la ubicación del usuario (Si el usuario se encuentra en el punto de origen el botón está deshabilitado)

función deshabilitarBotonAvanzar()

```
function desabilitarBotonAvanzar(){
  <?php if (count($puntos_navegacion) == 2):?>
    $("#popupDialog").popup("open",100,300);
    $('#link_avanzar').addClass('ui-disabled');
  <?php endif;?>
}
```

Función que deshabilita el botón avanzar de la barra de navegación dependiendo de de la ubicación del usuario (Si el usuario llegó al destino el botón es deshabilitado).

función deshabilitarBotonZoomIn()

Función que deshabilita el botón de ZoomIn dependiendo de si el usuario ha alcanzado el máximo de zoom permitido (El máximo permitido se encuentra parametrizado en la configuración de la aplicación).

función deshabilitarBotonZoomOut()

```
function desabilitarBotonZoomOut(){
  <?php if ($sf_user->getAttribute('escala') == sfConfig::get('app_maximo_escala')):?>
    $('#link_zoom_out').addClass('ui-disabled');
  <?php endif;?>
}
```

Función que deshabilita el botón de ZoomOut dependiendo de si el usuario ha alcanzado el mínimo de zoom permitido (El mínimo permitido se encuentra parametrizado en la configuración de la aplicación).

función centrar()

```
function centrar(){
    x = "<?php echo PuntoNavegacionTable::getPuntoNavegacionXPara($sf_user->getAttribute('actual_id'))
                                    / $sf_user->getAttribute('escala') + 10;?>";
    y = "<?php echo (sfConfig::get('app_maximo_y') -
                PuntoNavegacionTable::getPuntoNavegacionYPara($sf_user->getAttribute('actual_id'))) /
                $sf_user->getAttribute('escala');?>";

    offset = "<?php echo $sf_user->getAttribute('escala') ;?>";

    if(offset < 5){
      stage.setX( (-x/2) - (offset*50) )  ;
      stage.setY((-y/2) - (offset*50) );
    }else{
      stage.setX( (-x/2) + (offset*10)  )
      stage.setY((-y/2) + (offset*10) );
    }
    stage.draw();
}
```

Función que se encarga de centrar el canvas dependiendo del movimiento que va realizado el usuario en el mapa.

La siguiente pantalla es una vista del template navegarSucess - Basándonos en el usuario parado en la puerta de la facultad habiendo seleccionado como destino el Aula 2 (APL).

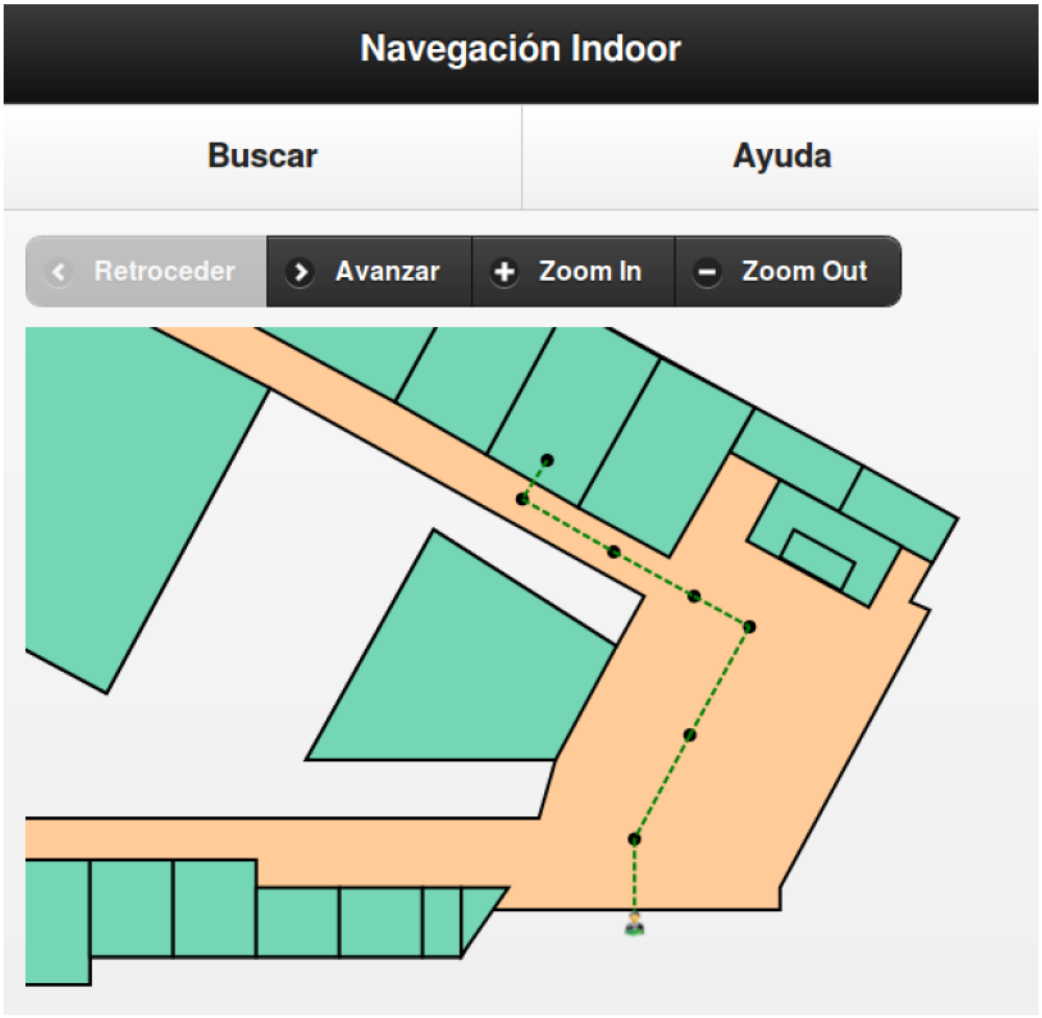


Figura 50. Pantalla template navegarSuccess